Il libro dell'

Gianni Becattini

Sergio Benini

Franco Pirri

Testo di carattere generale sul microprocessore F8 con particolare riferimento al sistema CHILD 8/BS della General Processor.

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA MONTEBELLO, 3 - 3a rosso
TEL. 055 / 219.143 — 50123 FIRENZE

I N T R O D U Z I O N E

Il presente lavoro è derivato dalla tesi di laurea del-l'ing. Sergio Benini. Tale lavoro è stato svolto sotto la guida dell'ing. Franco Pirri, relatore della tesi, e dello amico Gianni Becattini, titolare della General Processor di Firenze.

L'argomento della tesi era "Acquisizione di dati analogici tramite microelaboratore", con particolare riferimento ai segnali elettrocardiografici. Le numerose richieste pervenute hanno indotto a compiere una parziale revisione del lavoro, eliminando le parti inerenti alla conversione A/D ed alla acquisizione dati ed approfondendo invece le parti sul microprocessore F8, con lo scopo di fornire qualche appunto in lingua italiana sull'argomento.

La maggior parte delle notizie sono state derivate dalla documentazione Fairchild in lingua inglese cui rimandiamo il lettore per ulteriori informazioni e che rimane, in ogni caso e difficoltà di lingua a parte, la migliore base per completezza e chiarezza.

Gli Autori

CAPITOLO 1

Generalità sul Microprocessore

Un microprocessore è un componente a larga scala di integrazione controllato a programma, in grado di eseguire operazioni aritmetiche e logiche su dati binari.

La filosofia d'impiego di un microprocessore consiste nell'attuare un sistema in cui le funzioni logiche di un certo dispositivo non sono più realizzate da hardware ma da un programma inteso come sequenza di istruzioni memorizzate in un circuito che talora è una ROM (Read Only Memory) cioè un circuito di sola lettura.

Lo scopo di un sistema microprocessore è di rimpiazzare le logiche discrete, p.e. TTL ; si ha la possibilità, sostituendo il programma, di cambiare completamente la funzione eseguita dal microprocessore.

Il microprocessore si è inserito dunque fra il minicomputer e la logica cablata.

I principali campi di impiego sono:

- terminali per punti di vendita;
- controllori di unità periferiche di calcolatori;
- terminali intelligenti;
- unità di controllo per macchine utensili;
- controllori di processo;

- sistemi di telemisura;
- unità centrali di minicalcolatori;
- unità di gestione per reti di comunicazione;
- strumentazione.

Il successo che il microprocessore ha avuto nel mercato è dovuto al fatto che i costi sono sempre decrescenti, in rapporto alla sempre maggiore specializzazione, e si ha una maggiore rapidità di progetto e messa a punto, una volta acquisita l'esperienza minima necessaria: una buona parte di hardware può essere sostituita da una sequenza di istruzioni posta in memoria RAM (Ramdom Access Memory) memoria ad accesso casuale, durante la fase di sviluppo e successivamente in una ROM o PROM (Programmable Read Only Memory) cioè una memoria programmabile di sola lettura.

Il costo di un sistema digitale in funzione della sua struttura (logica cablata, microprocessore, minicalcolatore) e di due parametri caratteristici (complessità e velocità di elaborazione) può essere rappresentato da due curve caratteristiche (fig. 1:1).

Le curve mostrano le fasce applicative in cui i microprocessori sono più convenienti; queste corrispondono a basse velocità operative ed elaborazioni di media complessità.

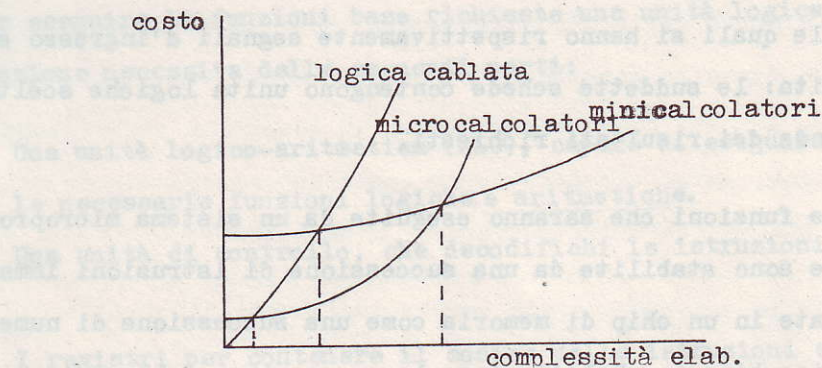
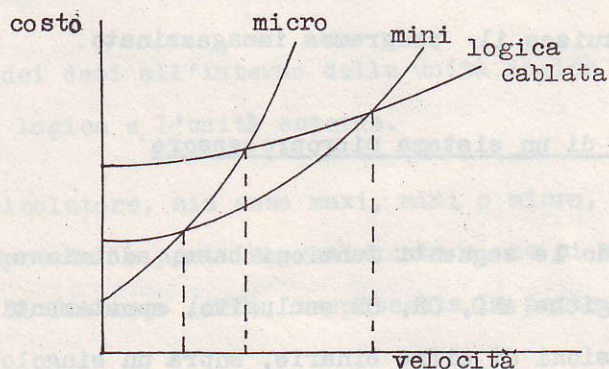



Fig 1.1

Inoltre un sistema a microprocessore presenta un'affidabilità superiore a quella di un sistema a logica cablata, richiedendo un minor numero di circuiti logici e di schede, e inoltre può subire delle modifiche sostanziali permettendo una immediata capacità di adeguamento alle richieste del mercato.

Il microprocessore è composto da una o più schede dai bor

di delle quali si hanno rispettivamente segnali d'ingresso e di uscita: le suddette schede contengono unità logiche scelte a seconda dei risultati richiesti.

Le funzioni che saranno eseguite da un sistema microprocessore sono stabilite da una successione di istruzioni immagazzinate in un chip di memoria come una successione di numeri in codice binario. Nell'insieme la successione delle istruzioni costituisce il programma immagazzinato.

Architettura di un sistema microprocessore

Eseguendo le seguenti funzioni base, addizione binaria, operazioni logiche AND, OR, OR esclusivo, spostamenti e rotazioni di successioni di cifre binarie, sopra un singolo chip, può essere realizzata una unità logica "general purpose".

All'interno di una unità logica entreranno in genere delle istruzioni e dei dati e usciranno dei dati, ciò si può schematizzare come in figura 1.2

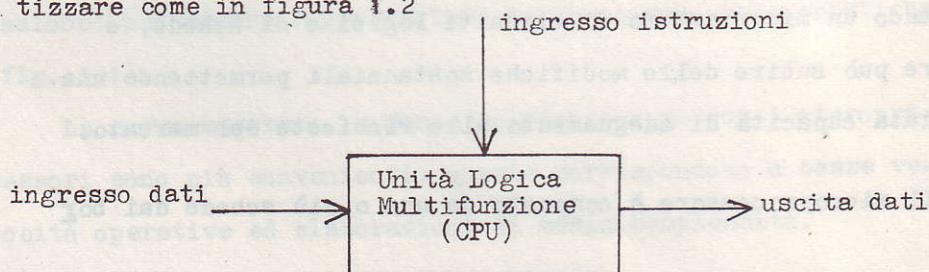



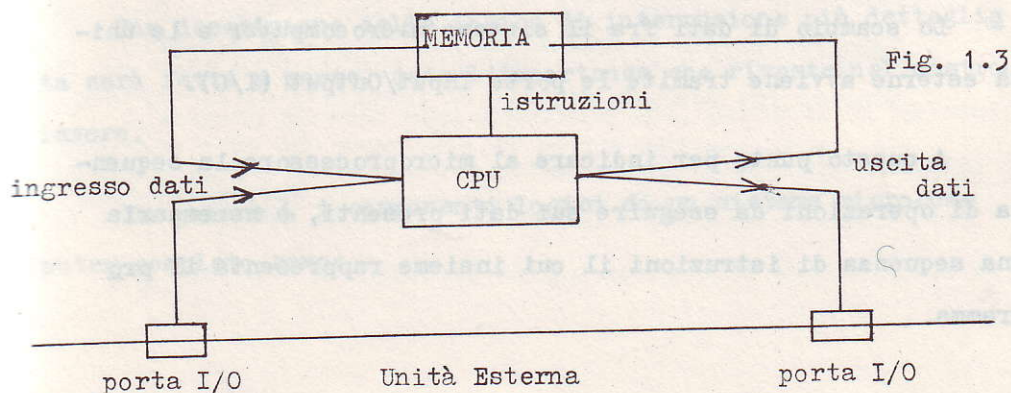
Fig 1.2

Per eseguire le funzioni base richieste una unità logica multifunzione necessita delle seguenti parti:

1) Una unità logico-aritmetica (ALU), capace di eseguire le necessarie funzioni logiche e aritmetiche.
2) Una unità di controllo, che decodifichi le istruzioni
3) I registri per contenere il codice delle istruzioni e i dati.
4) Le vie dei dati all'interno della unità logica e fra l'unità logica e l'unità esterna.

In ogni calcolatore, sia esso maxi, mini o micro, l'unità logica, che comprende le parti sopraelencate e che si chiama Unità Centrale di Processo (CPU), rappresenta la parte più importante.

Lo schema di figura 1.2 va ampliato prendendo in considerazione la sorgente delle istruzioni e dei dati e gli eventuali utilizzatori dei dati elaborati

Fig. 1.3

La parte della figura indicata con il nome di memoria indica un deposito passivo di informazioni.

La memoria deve essere divisa in locazioni indirizzabili individualmente, ciascuna delle quali può immagazzinare un elemento del codice delle istruzioni o un elemento dei dati.

Nella memoria possono essere registrati diversi tipi di informazioni:

1) Il codice di una istruzione che può occupare più locazioni consecutive.
2) Una variabile numerica o alfabetica.
3) Una costante.

Nella figura 1.3 compare anche una parte indicata come "Unità Esterne": si indicano con questo nome le sorgenti o destinazioni di dati al di fuori del sistema microcomputer. Facendo una analogia con la scheda logica possiamo dire che le unità esterne si riferiscono al mondo al di la del bordo della scheda.

Lo scambio di dati fra il sistema microcomputer e le unità esterne avviene tramite le porte Input/Output (I/O).

A questo punto per indicare al microprocessore la sequenza di operazioni da eseguire sui dati presenti, è necessaria una sequenza di istruzioni il cui insieme rappresenta il programma.

In genere una istruzione sarà rappresentata in memoria in due parti: il Codice Operativo e l'Operando .

Codice Operativo	Operando

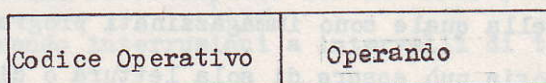



Fig. 1.4

Il codice operativo indica al microprocessore cosa deve fare all'arrivo di quella particolare istruzione, cioè aprire o chiudere certe porte logiche in quanto quella certa sequenza di bit agisce sull'hardware facendo lavorare la circuiteria logica in un certo modo.

L'operando rappresenta il dato o l'indirizzo del dato (locazione di memoria o registro) su cui si vuole operare con la istruzione.

Ogni microcalcolatore, oltre alle parti già descritte possiede altri elementi utili allo svolgimento di funzioni complesse. C'è una logica di interruzione che serve per interrompere lo svolgimento del programma corrente , quando si verificano eventi particolari esterni, e che comanda lo svolgimento di altri programmi utili in quelle situazioni.

Una descrizione della logica di interruzione più dettagliata sarà fatta a parte, data l'importanza che riveste nel nostro lavoro.

In genere , i componenti logici di un sistema microcomputer completo sono:

1) Una CPU, che è l'unità logica multifunzione del sistema.

2) La memoria nella quale sono immagazzinati programmi e dati: la memoria può essere di sola lettura o di lettura/scrittura.

3) Un interfaccia logica per le memorie, che identifica:
   a) La prossima locazione di memoria che contiene il codice dell' istruzione per la CPU.
   b) La locazione di memoria dalla quale un byte di dati sarà letto, o nella quale un byte di dati sarà scritto.

4) Porte I/O bidirezionali, attraverso le quali i dati passano fra il microprocessore e l'unità esterne.

5) Logica DMA (Direct Memory Access), che fornisce un canale diretto per il flusso di dati fra la memoria e le unità esterne, saltando la CPU.

6) Logica di interruzione, che fa sì che la CPU sospenda temporaneamente l'esecuzione del programma corrente. Insieme con ciascun segnale di richiesta di interruzione, la logica identifica il programma che deve compiere le operazioni richieste dalla sorgente dell'interruzione.

7) Un clock in tempo reale, che sincronizza l'intero sistema microcomputer con il mondo reale esterno, generando interruzioni a intervalli di tempo variabili definiti dal programma.

Fra i componenti logici di un sistema microcomputer compaiono anche i Clocks Programmabili e il DMA: descriviamoli più dettagliatamente.

I Clocks Programmabili sono dei registri il cui contenuto è decrementato a frequenza nota: quando il contenuto del registro è zero, l'evento viene segnalato da un'interruzione chiamata in questo caso interruzione di "time out".

I Clocks sono molto importanti in molte applicazioni dei microcomputer, infatti permettono di sincronizzare il microcomputer con il tempo reale del mondo esterno.

Abbiamo visto che i dati per essere messi in memoria, partendo dalle unità esterne, devono passare attraverso la CPU. Così facendo si crea un vincolo alla logica della CPU che deve smistare i dati da una porta I/O alla memoria, può essere quindi talvolta preferibile accedere direttamente alla memoria.

L'accesso diretto alla memoria o DMA, permette di spostare direttamente i dati dalla porta I/O alla memoria, saltando la CPU. La porta ingresso/uscita DMA è chiamata "canale DMA".

Un corretto svolgimento di una operazione DMA richiede che

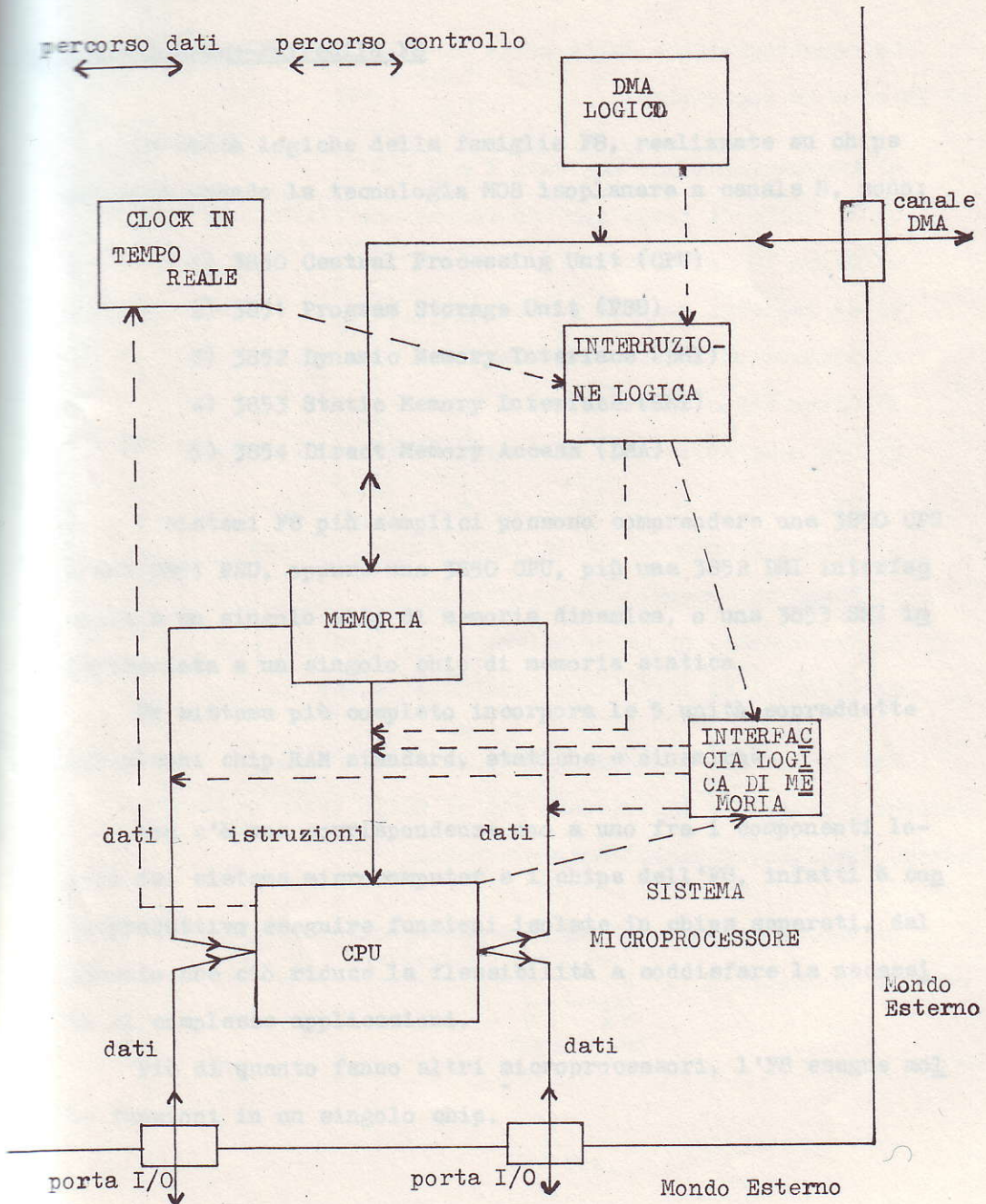
il microcomputer abbia le seguenti informazioni:

1) Un indirizzo di inizio nella memoria del blocco di dati da trasferire.

2) La lunghezza in byte del blocco dei dati.

3) La direzione del movimento dei dati.

Fig. 1.5

percorso dati
percorso controllo
DMA LOGICO
canale DMA
CLOCK IN TEMPO REALE
INTERRUZIONE LOGICA
MEMORIA
INTERFACCIA LOGICA DI MEMORIA
dati
istruzioni
dati
SISTEMA MICROPROCESSORE
CPU
Mondo Esterno
dati
dati
porta I/O
porta I/O
Mondo Esterno

CAPITOLO 2

Microprocessore Fairchild F8

Le unità logiche della famiglia F8, realizzate su chips prodotti usando la tecnologia MOS isoplanare a canale N, sono:

1) 3850 Central Processing Unit (CPU)

2) 3851 Program Storage Unit (PSU)

3) 3852 Dynamic Memory Interface (DMI)

4) 3853 Static Memory Interface (SMI)

5) 3854 Direct Memory Access (DMA)

I sistemi F8 più semplici possono comprendere una 3850 CPU e una 3851 PSU, oppure una 3850 CPU, più una 3852 DMI interfacciata a un singolo chip di memoria dinamica, o una 3853 SMI interfacciata a un singolo chip di memoria statica.

Un sistema più completo incorpora le 5 unità sopraddette più alcuni chip RAM standard, statiche e dinamiche.

Non c'è una corrispondenza uno a uno fra i componenti logici del sistema microcomputer e i chips dell'F8, infatti è controproduttivo eseguire funzioni isolate in chips separati, dal momento che ciò riduce la flessibilità a soddisfare la necessità di complesse applicazioni.

Più di quanto fanno altri microprocessori, l'F8 esegue molte funzioni in un singolo chip.

Le caratteristiche della serie di chips del microprocessore F8 sono le seguenti:

a) L'organizzazione dei dati è a 8 bit
b) Il ciclo di tempo per una istruzione (di 8 bit) è 2µS
c) Oltre 70 istruzioni
d) 64 registri general purpose di 8 bit
e) Aritmetica binaria e decimale, e funzioni logiche
f) Oltre 65536 bytes di ROM e RAM
g) Non sono necessari chips d'interfaccia per unità esterne
h) Clocks programmabili in tempo reale, interni
i) Power on reset interno
l) Trattamento di interruzione a più livelli
m) Circuiti di clock e di sincronizzazione

Per quello che riguarda quanto detto in f) si può aggiungereche è possibile indirizzare più di 65536 bytes di memoria usando tecniche speciali.

La configurazione del sistema microprocessore F8 è rappresentata in figura 2.1.

3850
CPU
I/O
I/O
Ø
7
3851
PSU
3853
SMI
Memoria statica di Lett./Scrit.
3852
DMI
Memoria dinamica di Lett./Scrit.
Controllo
3854
DMA
0
Canale DMA
VIE DATI E CONTROLLO

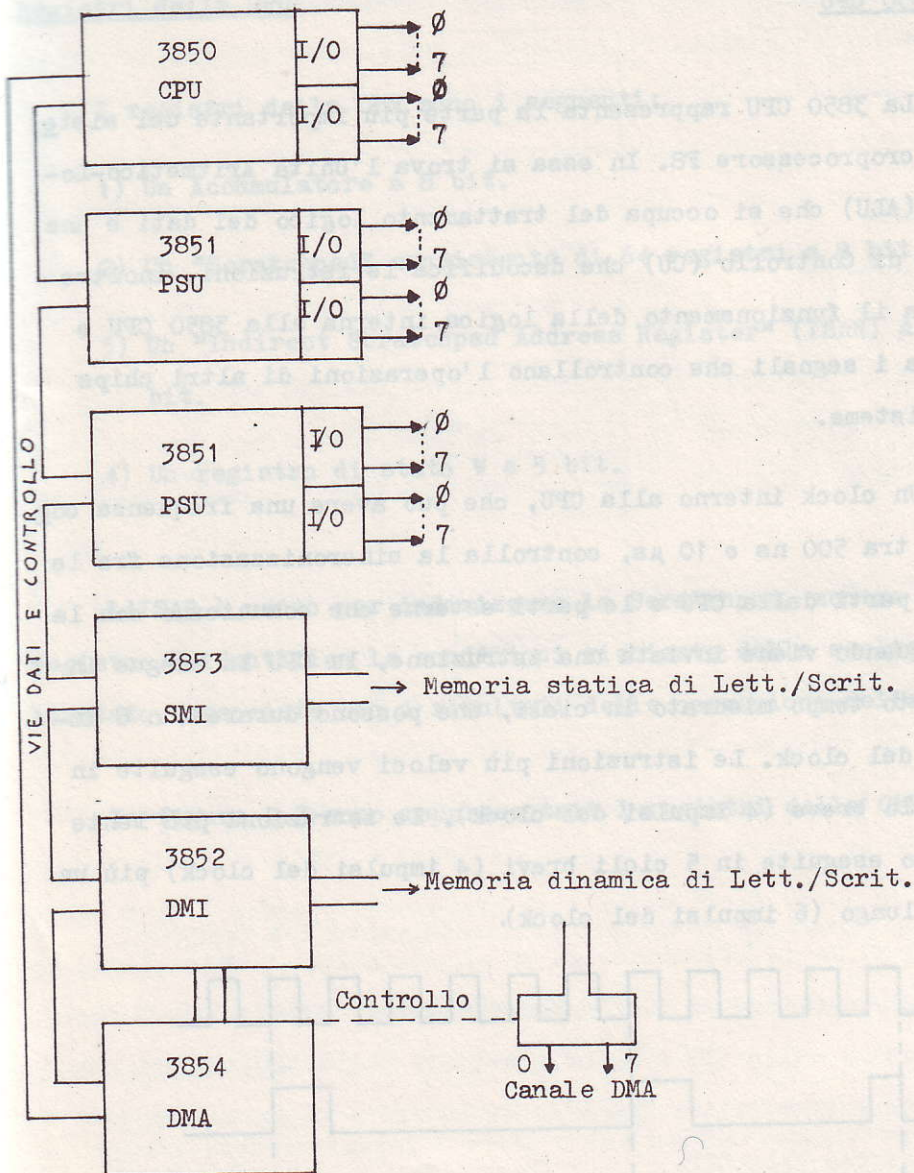



Fig. 2.1

La 3850 CPU

La 3850 CPU rappresenta la parte più importante del sistema microprocessore F8. In essa si trova l'Unità Aritmetico-Logica (ALU) che si occupa del trattamento logico dei dati e una Unità di Controllo (CU) che decodifica le istruzioni, inoltre regola il funzionamento della logica interna alla 3850 CPU e genera i segnali che controllano l'operazioni di altri chips nel sistema.

Un clock interno alla CPU, che può avere una frequenza compresa tra 500 ns e 10 µs, controlla la sincronizzazione fra le varie parti della CPU e le parti esterne che comunicano con la CPU. Quando viene inviata una istruzione, la CPU la esegue in un certo tempo misurato in cicli, che possono durare 4 o 6 impulsi del clock. Le istruzioni più veloci vengono eseguite in un ciclo breve (4 impulsi del clock), le istruzioni più lente vengono eseguite in 5 cicli brevi (4 impulsi del clock) più un ciclo lungo (6 impulsi del clock).

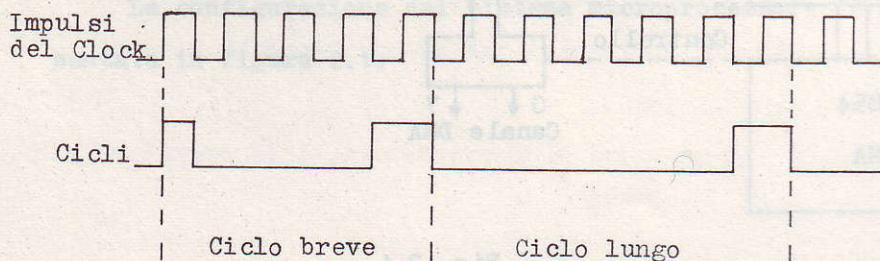



Fig. 2.2

Registri della CPU

I registri della CPU sono i seguenti:

1) Un Accumulatore a 8 bit.

2) Un "Scratchpad" consistente di 64 registri a 8 bit.

3) Un "Indirect Scratchpad Address Register" (ISAR) a 6 bit.

4) Un registro di stato W a 5 bit.

L'ISAR è usato per indirizzare lo Scratchpad, mentre il registro W identifica le condizioni richieste dalla scelta dello stato, associate con i risultati delle operazioni della CPU.

In figura 2.3 sono rappresentati i registri della CPU.

ACCUMULATORE
7 6 5 4 3 2 1 0

ISAR
5 4 3 2 1 0
HI LO
Indirizzo ottale del byte Scratchpad

REGISTRO W
4 3 2 1 0
ICB O Z C S

SCRATCHPAD

Registro	Byte	Byte Indirizzo Decimale	Byte Indirizzo Ottale
		0	0
		1	1
		2	2
		3	3
	J	9	11
H	HU	10	12
	HL	11	13
K	KU	12	14
	KL	13	15
Q	QU	14	16
	QL	15	17
		16	20
		17	[illegible]
		61	74
		62	75
		63	76
		64	77

Fig. 2.3

Analiziamo uno a uno più particolarmente i 4 registri.

Dall'Accumulatore i dati possono essere trasferiti ad altri registri della CPU e quindi trattati dalla ALU, oppure possono essere trasferiti fra l'Accumulatore e locazioni di memoria esterne alla CPU.

La Scratchpad nelle configurazioni dei piccoli microcalcolatori può rappresentare l'unica memoria di lettura/scrittura del sistema: questo registro è il deposito di dati a cui si accede più frequentemente. Inoltre le istruzioni che si riferiscono allo Scratchpad sono eseguite in un ciclo breve, e sono quindi le istruzioni più veloci. I registri da 10 a 15 sono connessi all'interfaccia logica di memoria.

L'ISAR, come abbiamo detto, serve per indirizzare lo Scratchpad, si può aggiungere però che i primi 16 registri dello Scratchpad possono essere identificati anche senza l'uso dell'ISAR.

Si può dividere il contenuto dell'ISAR in due cifre ottali, i bit di ordine alto HI e quelli di ordine basso LO. Le istruzioni che arrivano alla CPU incrementano o decrementano il contenuto dell'ISAR, o meglio ciò che viene incrementato o decrementato è la cifra LO. Perciò se si incrementa l'ISAR per esempio si va da O'27' a O'20' e non a O'30', infatti se 010 111 (= O'27') viene incrementato di 1 nella cifra bassa si ha 010 000 (= O'20'), analogamente se si decrementa si va da O'20' a O'27' e non a O'17' (la parte bassa, cioè 0, meno 1 è 000 più il complemento a 2 di 001, cioè 110+).
$\overline{111}$

Questa caratteristica è molto utile perchè semplifica una serie di programmi.

Il Registro W, indica lo stato del sistema dopo che la ALU ha eseguito una qualunque istruzione, inoltre è collegato al registro 9 della Scratchpad, quindi i dati possono passare direttamente fra il registro 9 e il Registro W saltando l'Accumulatore.

Analiziamo più in dettaglio i 5 bit del Registro W.

SIGN Il bit indicato con S (bit 0) riguarda il segno del numero che si ottiene svolgendo una certa operazione nell'ALU: il segno è rappresentato dal settimo degli 8 bit. Si ha che il bit S è posto al complemento del bit 7 dell'Accumulatore.

CARRY Il bit indicato con C (bit 1) si può vedere come il nono bit di una unità dati a 9 bit, infatti il bit C ci rappresenta il riporto di una somma maggiore di 255.

Esempio senza riporto; C è azzerato.

```
C  76543210     Numero del bit
    01100101    Contenuto Accumulatore
    01110110    Valore aggiunto
   _________
   011011011
```

Esempio con riporto; C è posto a 1.

```
C  76543210
    10011101
    11010001
   _________
   101101110
```

ZERO Il bit indicato con Z (bit 2) è posto a 1 quando un'operazione logica o aritmetica da risultato zero, è azzerato quando l'operazione avrebbe potuto generare un risultato zero, ma ciò non si è verificato.

Esempio con bit Z azzerato (e anche C azzerato):

```
C  01101011
 \ 00010101
 010000000
```

Ora se il contenuto dell'Accumulatore è shiftato a sinistra di una posizione, il bit Z è posto a 1.

Molte istruzioni non modificano il registro di stato W.

OVERFLOW Il bit indicato con O (numero 3) considera il riporto nel bit 6 dell'Accumulatore. Dopo una operazione aritmetica il bit O assume il valore dell'operazione logica OR esclusivo fra il riporto del bit 6 e il riporto del bit 7.

Esempio con il bit O posto a O (e il bit C posto a 1):

```
C  10110011
 \ 01110001
 100100100
```

Il riporto del bit 6 è 1, come lo è quello del bit 7, allora $1 \oplus 1 = 0$ $\longrightarrow$ il bit O posto a O.

Esempio con il bit O posto a 1 (e il bit C azzerato):

```
C  01100111            0 (+) 1=1
 \ 00100100
 010001011
```

ICB Il bit indicato con ICB (numero 4) abilita o disabilita le interruzioni, rispettivamente se è posto a 1 o se è azzerato. (Interrupt Control Bit)

SOMMARIO DEI BIT DI STATO

Overflow = $carry_7 \oplus carry_6$

Zero = $\overline{ALU_7} \wedge \overline{ALU_6} \wedge \overline{ALU_5} \wedge \overline{ALU_4} \wedge \overline{ALU_3} \wedge \overline{ALU_2} \wedge \overline{ALU_1} \wedge \overline{ALU_0}$

Carry = $carry_7$

Sign = $\overline{ALU_7}$

3850 Input/Output

La Unita' Centrale 3850 CPU ha due porte bidirezionali a 8 bit, attraverso le quali i dati possono essere trasferiti parallelamente ed in modo bidirezionale, fra la 3850 CPU ed una logica esterna al sistema microprocessore. Le due porte di ingresso/uscita della 3850 CPU sono identificate dagli indirizzi in esadecimale H'00' e H'01'.

La 3851 PSU

Il chip 3851 PSU contiene una memoria di sola lettura ROM con 1024 bytes e 3 registri a 16 bit. Inoltre ciascuna 3851 PSU ha anche due porte I/O, un timer programmabile e una logica di processo di interruzioni esterne.

La memoria 3851 è generalmente usata per immagazzinare le istruzioni ed è proprio questa memoria che sarà modificata o sostituita quando si vuole modificare la funzione di

un dispositivo a microprocessore gia' in produzione. Oltre ai codici delle istruzioni nella memoria 3851 possono essere memorizzati anche i dati in sola lettura

Un singolo chip 3851 PSU, interfacciato con un chip 3850 CPU, costituisce un sistema avente le seguenti capacità:

1) 1024 bytes di immagazzinamento del programma (nella 3851).

2) 64 bytes di memoria di lettura/scrittura (nella 3850).

3) 4 porte I/O bidirezionali, separatamente indirizzabili (2 nella 3850 e 2 nella 3851).

4) Una linea esterna di interruzione.

5) Un clock programmabile.

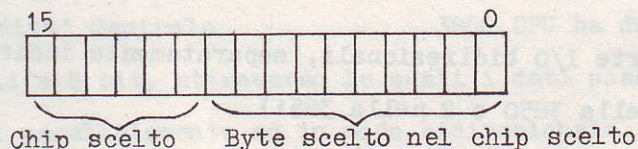
La sincronizzazione delle operazioni della PSU è eseguita dal clock generato della CPU.

I registri della PSU

Come abbiamo già accennato la 3851 PSU contiene 3 registri a 16 bit e precisamente il Program Counter (PCO), lo Stack Register (PC1) e il Data Counter (DC). Analiziamoli uno per uno.

Il Program Counter (PCO) è un registro che contiene l'in-

dirizzo di memoria dell'istruzione, seguente a quella che è in corso, che verrà poi trasmessa alla CPU. Dopo che l'istruzione, puntata del PCO, è stata trasmessa, il contenuto del PCO viene incrementato automaticamente per indirizzare la prossima posizione di memoria.

Benchè una 3851 PSU contenga 1024 bytes di memoria, il registro PCO conserva un indirizzo a 16 bit di memoria. Un indirizzo del PCO può essere interpretato come segue:

Si può accedere alla memoria 3851 solamente quando gli ultimi 6 bit dell'indirizzo, che indicano il chip scelto, uguagliano il codice di scelta della 3851 PSU: questo codice di scelta, a richiesta, può essere scritto permanentemente direttamente in fabbrica.

Si ha così che, se in un sistema microcomputer F8 sono presenti più di un chip 3851 PSU, tutti conterranno il prossimo indirizzo di memoria, che contiene l'istruzione da inviare alla CPU, ma solamente il chip, il cui codice di scelta corrisponde con gli ultimi 6 bit, sarà attivato.

I registri PCO dei chips 3851 sono logicamente connessi ai

registri 12 e 13, 14 e 15, indicati con K, Q dello Scratchpad, quindi si hanno delle istruzioni che permettono di caricare i contenuti del registro K o Q in tutti i registri PCO. Ci sono anche altre istruzioni che permettono che i contenuti del registro PCO siano modificati al fine di controllare le sequenze logiche del microprocessore.

Molto importante è anche lo Stack Register (PC1), un registro temporaneo a 16 bit per il contenuto del PCO. Quando durante l'esecuzione di un programma viene chiamata una subroutine viene cambiato il contenuto di PCO e per questo una volta eseguita la subroutine sarebbe impossibile tornare al programma principale. Si può ovviare a questi inconvenienti salvando il contenuto del PCO prima dell'interruzione del programma principale nello Stack Register.

Come il PCO anche i registri PC1 sono logicamente connessi ai registri 12 e 13, e ci sono anche specifiche istruzioni che permettono di caricare i contenuti del registro K in PC1 o viceversa.

Si può dire infine che, mentre il contenuto di PCO è salvato in PC1, il contenuto dello Stack Register è salvato in K.

Ultimo registro della PSU è il Data Counter (DC), registro a 16 bit che svolge le funzioni di: "puntatore dati". Il DC contiene l'indirizzo della posizione di memoria, esterno alla 3850

CPU, in cui si trovano i dati a cui si vuole accedere. Quello che viene puntato dal DC per esempio sarà portato dalla memoria esterna all'Accumulatore.

Dei 16 bit contenuti nel DC, come per il PCO, quelli di ordine più alto (da 10 a 15) indicano il chip scelto, gli altri di ordine basso (da 0 a 9) stabiliscono l'indirizzo del byte all'interno del chip.

Si ha inoltre che i registri DC sono logicamente collegati ai registri Q e H dello Scratchpad della CPU.

<u>3851 Input/Output, Timer e Interruzioni Locali</u>

La 3851 PSU ha due porte I/O bidirezionali a 8 bit. Usando la notazione binaria gli indirizzi di ciascuna porta si possono rappresentare così:

XXXXXX00 e XXXXXX01

Le cifre binarie X rappresentano il codice di scelta della porta I/O, mentre gli altri due bit rappresentano un codice indipendente di scelta del chip.

Se il codice di scelta della porta I/O è 000000, allora le 2 porte I/O non possono essere indirizzate, poichè le porte I/O 3850 usano gli indirizzi binari 00000000 e 00000001. Perciò il codice binario 000000 non è usato come codice discelta delle porte I/O di una 3851 PSU.

Il timer programmabile 3851 e la logica dell'interruzione sono accessibili tramite gli indirizzi binari XXXXXX11 e XXXXXX10. (Per la PSU col debug valgono Ø7 e Ø6)

Il timer programmabile è un registro che manda un segnale alla logica di controllo delle interruzioni ad intervalli prefissati.

Caricati i valori numerici compresi fra 0 e 255 con un appropriato codice , il timer li decrementa ogni 31 impulsi del clock: se si ha 255 il timer si ferma.

Le interruzioni locali sono comandate da un'appropriata istruzione, con un codice di controllo: vediamo come sono interpretati i bit, zero e uno del codice di controllo:

bit 1	bit 0	Funzione
0	0	Disabilita tutte le interruzioni
0	1	Abilita le interruzioni esterne
1	0	Disabilita tutte le interruzioni
1	1	Abilita le interruzioni del timer

Se le interruzioni del timer sono state abilitate e se la 3850 CPU aveva abilitato le interruzioni (attraverso il bit di stato ICB), allora quando il timer locale raggiunge il valore 0, una richiesta di interruzione è trasmessa alla 3850 CPU.

La 3852 Dynamic Memory Interface

Il chip 3852 DMI interfaccia 65536 bytes di memoria RAM alla 3850 CPU. Se al sistema microcomputer è aggiunta una logica che interfaccia una speciale memoria addizionale si può andare oltre i 65536 bytes. In un sistema microcomputer F8 è presente al più un solo chip 3852 DMI.

Si può unire la 3852 DMI alla 3854 DMA, abilitando così il trasferimento di dati fra la memoria e le unità esterne, senza passare attraverso la CPU.

I registri della 3852 DMI

Questi registri sono: il Program Counter (PCO), lo Stack Pointer (PC1) e due Data Counters (DCO e DC1).

Una differenza nell'uso dei registri della 3852 DMI, rispetto alla 3851 PSU, consiste nel fatto che la 3852 non ha un codice fissato di scelta del chip, questo perchè deve essere presente un solo chip 3852 in un sistema F8 e l'intero indirizzo PCO è quindi trasmesso nel blocco di RAM comandato dalla 3852.

Premesso che il Data Counter DC1 è un registro di immagazzinamento temporaneo per il DCO, e non avendo la 3851 PSU un DC1, si può dire che una istruzione che sposta i contenuti del

DCO e del DC1 non ha effetto sul chip 3851 PSU. E' così possibile per il DCO della 3852 DMI avere contenuti che differiscono dai contenuti del DC della 3851 PSU.

Anche i Data Counter della 3852 DMI sono logicamente connessi, come quello della PSU, ai registri H e Q dello Scratchpad e il fatto che i Data Counters della DMI e della PSU possono avere un contenuto diverso non comporta alcun problema dal momento che il contenuto di un solo Data Counter è trasferito ai registri Q o H.

Si hanno inoltre nella 3852 DMI due porte indirizzabili, i cui indirizzi in esadecimale sono H'EC' e H'ED', che sono usate per abilitare il trasferimento diretto dei dati fra i chip di memoria e l'unità esterne: questo tipo di trasferimento richiede la presenza del chip 3854 DMA.

Per la porta H'EC' i primi tre bit hanno il seguente significato:

N. bit		
0	Se è a 1 ⇒ DMA in uso	Se è a 0 ⇒ DMA chiuso
1	Se è a 1 = "refresh" della memoria	
	Se è a 0 = non "refresh" della memoria	
2	Se è a 1 = "refresh" ogni 4 cicli di scrittura	
	Se è a 0 = " " 8 " " "	

<u>La 3853 Static Memory Interface</u>

La 3853 SMI è simile alla 3852 DMI; tuttavia 4 importanti differenze che sono di seguito elencate:

1) La 3853 SMI non ha la capacità d'interfaccia con il DMA.

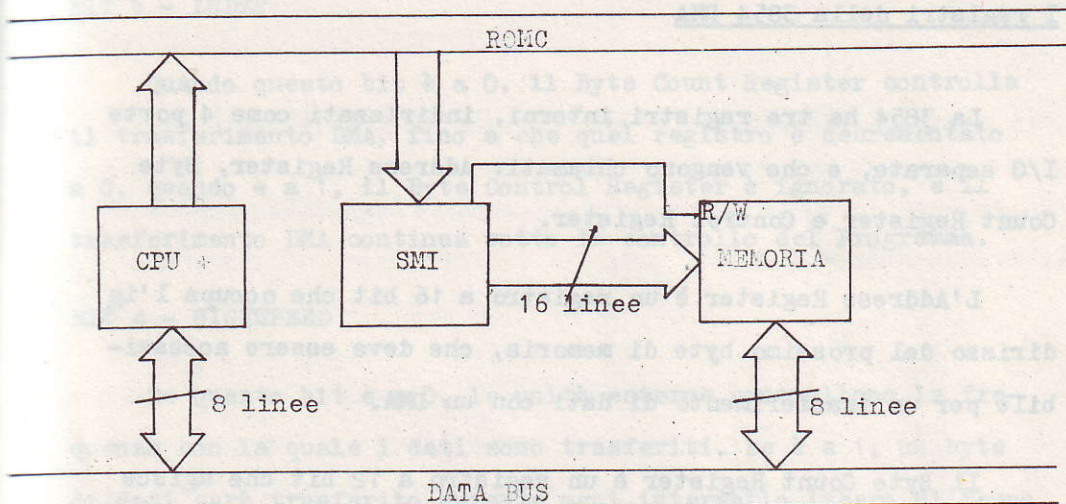
2) La 3853 SMI ha un timer locale e un'interruzione di controllo, i cui indirizzi sono rispettivamente H'OF' e H'OE'.

3) La 3853 SMI ha due porte addizionali, indirizzate con H'OC' e H'OD' che sono registri di vettori d'interruzione programmabili.

4) Il chip 3853 SMI interfaccia una memoria statica alla 3850 CPU.

In un sistema microprocessore F8 non ci possono essere più di una 3852 DMI e una 3853 SMI.

La CPU, tramite la Static Memory Interface (vedi fig. 2.4), può indirizzare, attraverso 16 linee, 65536 posizioni di memoria (16 linee $\longrightarrow 2^{16}$ =65536 combinazioni diverse).

Fig. 2.4

La 3854 Direct Memory Access

Il 3854 DMA, in unione con il 3852 DMI, permette ai dati di passare direttamente fra una porta I/O e una memoria del sistema microprocessore F8: il trasferimento dei dati attraverso il DMA avviene durante il secondo o il terzo impulso del clock di ciascun ciclo di istruzioni.

Ci possono essere fino a 4 chips 3854 DMA in un sistema microcomputer.

Un'unità esterna può essere unita al chip 3854 DMA e inoltre due sistemi microprocessori possono comunicare fra loro attraverso il chip DMA.

I registri della 3854 DMA

La 3854 ha tre registri interni, indirizzati come 4 porte I/O separate, e che vengono chiamati: Address Register, Byte Count Register e Control Register.

L'Address Register è un registro a 16 bit che occupa l'indirizzo del prossimo byte di memoria, che deve essere accessibile per un trasferimento di dati con un DMA.

Il Byte Count Register è un registro a 12 bit che agisce come un contatore, permettendo che i blocchi di oltre 4096 bytes di dati siano trasferiti durante una operazione DMA. Via via che un byte di dati è trasferito il Byte Count Register è decrementato e quando è a zero il trasferimento dei dati cessa.

Il Control Register è un registro a 4 bit che controlla le operazioni DMA come segue:

BIT 7 - ENABLE

Questo bit se è a 1 inizializza le operazioni DMA, e va a 0 automaticamente quando le operazioni sono finite.

BIT 6 - DIRECTION

Se questo bit è a 0, i dati sono trasferiti dalla memoria principale all'unità esterna, se è a 1 si ha il viceversa.

BIT 5 - INDEF

Quando questo bit è a 0, il Byte Count Register controlla il trasferimento DMA, fino a che quel registro è decrementato a 0, quando è a 1, il Byte Control Register è ignorato, e il trasferimento DMA continua sotto il controllo del Programma.

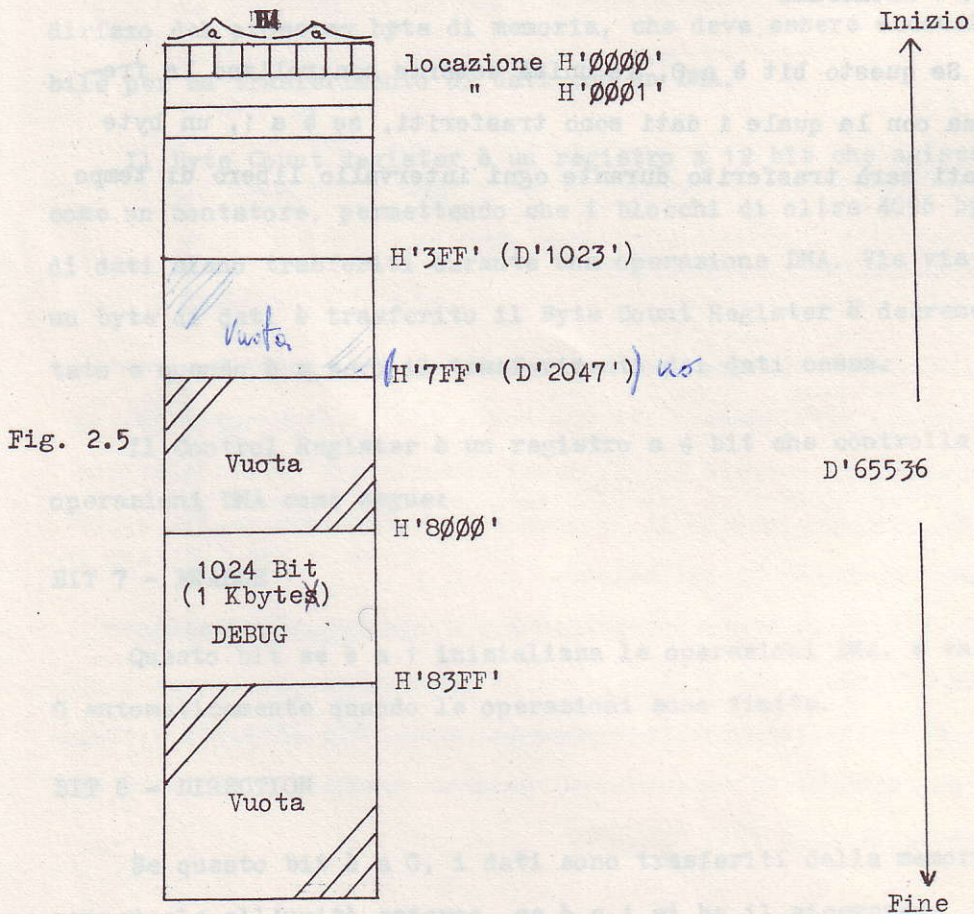
BIT 4 - HIGHSPEED

Se questo bit è a 0, le unità esterne controllano la frequenza con la quale i dati sono trasferiti, se è a 1, un byte di dati sarà trasferito durante ogni intervallo libero di tempo DMA

Memoria F8

Nei microcalcolatori della serie F8 la memoria e' ad 8 bit ed ha una capacità massima di 64 K bytes.

In figura 2.5 e' rappresentata la memoria del CHILD 8/BS nella configurazione minima.

Fig. 2.5

Le espansioni in blocchi di 4 K possono essere allocate in qualun qualunque zona si desideri con alcune predisposizioni.

CAPITOLO 3

Programmazione microprocessore F8

Le istruzioni per l'F8 sono piu' di 70 e possono essere divise in 10 categorie: Accumulator, Scratchpad Register, Indirect Scratchpad Register, Memory Reference, Data Counter, Status Register, Program Counter, Branch, Interrupt Control e Input/Output a seconda della loro funzione.

La durata di queste istruzioni si misura in cicli: ciascun ciclo dura 2 µS per un sistema con il clock a 2 MHz.

La programmazione in linguaggio macchina, ossia usando i codici esadecimali di ogni istruzione (programma "oggetto"), presenta alcune difficolta' di ordine mnemonico. Pertanto sono stati creati speciali programmi per tradurre un linguaggio di tipo piu' facilmente comprensibile, e piu' vicino al linguaggio umano, nei relativi codici esdaecimali. Il linguaggio piu' usato per la programmazione dei microcomputer e' il cosiddetto ASSEMBLER. Il programma scritto in assembler si chiama "sorgente".

Le istruzioni che compongono il programma sorgente possono essere di tre tipi: di commento, eseguibili, direttive assemblatrici.

Le prime servono per fare delle note nel programma, in modo da poterlo meglio identificare, mentre le seconde sono istruzioni che indicano i passi per implementare lo svolgimento del programma.

Le direttive assemblatrici invece non generano alcun codice oggetto, ma piuttosto forniscono informazioni addi-zionali intorno al modo di assemblare il programma, per esempio possono dare la base di numeraz-ione per le costanti, oppure possono indicare la locazione dove deve essere posto il programma in memoria.

NOTA: Si rimanda il lettore, per l'argomento linguaggi di programmazione ai numerosi testi, anche in lingua italiana, facilmente reperibili.

Le istruzioni eseguibili e le direttive assemblatrici sono composte da quattro parti:

1) Label field
2) Mnemonic field
3) Operand field
4) Comment field

Il LABEL FIELD serve per assegnare un nome ad una specifica istruzione o ad un insieme di istruzioni: può essere usato qualsiasi simbolo.

Il MNEMONIC FIELD contiene il Codice Operativo, che identifica l'operazione che deve essere eseguita.

L'OPERAND FIELD è composto da informazioni aggiuntive, per esempio parametri, indirizzi ecc., richieste dall'Assemblatore per identificare perfettamente l'operazione da compiere: l'operand field può contenere un simbolo o un'espressione.

Infine il COMMENT FIELD è opzionale e serve per aggiungere informazioni che rendono più chiaro il programma.

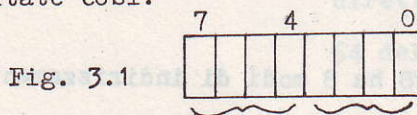
Rappresentazione delle istruzioni

Le istruzioni possono essere rappresentate con 1, 2, 3 bytes del codice oggetto. Nel primo byte è sempre contenuto il codice operativo delle istruzioni. Il secondo byte, quando sono presenti due bytes, rappresenta numeri binari con segno o senza segno. Il secondo e terzo bytes di una istruzione di tre bytes rappresenta numeri binari a 16 bit senza segno.

Da notare che esistono delle istruzioni che possono essere rappresentate da 1 o 2 bytes del codice oggetto a seconda della lunghezza dell'operando: in questi casi le istruzioni di 1 solo byte vengono caratterizzate dalla scritta "Short". Si ha, per esempio, OUTPUT e OUTPUT SHORT e ancora LOAD IMMEDIATE e LOAD IMMEDIATE SHORT.

Le istruzioni di un solo byte, come LIS, CLR ecc., sono rappresentate così:

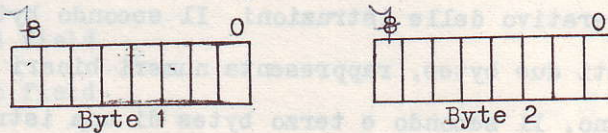
7			4				0

Fig. 3.1

I 4 bit da 0 a 3 contengono un numero binario senza segno che può rappresentare il registro, una porta I/O, oppure un dato semplice, gli altri bit da 4 a 7 rappresentano il codice dell'istruzione.

Si possono avere sempre istruzioni di un solo byte ma con

una suddivisione diversa. Per esempio LISU ha 5 bit delle istruzioni e 3 bit per un numero senza segno, mentre DI, EI, ST ecc. hanno 8 bit per il codice dell'istruzione.

Per le istruzioni di due bytes si ha questa configurazione:

Fig. 3.2

Il primo byte rappresenta il codice dell'istruzione, il secondo però rappresenta un dato senza segno o con segno: vale il primo caso per IN, LI, OUT ecc., il secondo per BNZ, BZ, BNO ecc.

Le istruzioni di tre bytes sono DCI, JMP, PI: il primo contiene il codice delle istruzioni, gli altri due un dato senza segno di 16 bit.

Modi di indirizzamento per l'F8

Il set di istruzioni F8 ha 8 modi di indirizzamento, per l'ingresso/uscita, per i registri della CPU e per il blocco di memoria.

1) IMPLIED ADDRESSING: il riferimento di questo tipo di istruzione è implicito nell'istruzione stessa.

2) DIRECT ADDRESSING: in queste istruzioni, l'indirizzo dell'operando è contenuto nel secondo byte del l'istruzione. Il Direct Addressing è usato nella classe di istruzioni Input/Output.

3) SHORT IMMEDIATE ADDRESSING: le istruzioni di un solo byte, che hanno questo tipo di indirizzamento contengono il codice operativo nei primi 4 bit e lo operando negli ultimi 4.

4) LONG IMMEDIATE ADDRESSING: queste istruzioni di due byte contengono, nel primo, il codice operativo, nel secondo, l'operando.

5) DIRECT REGISTER ADDRESSING: questo metodo di indirizzamento può essere usato per riferirsi direttamente a 12 registri dei 64 dello Scratchpad Register.

6) INDIRECT REGISTER ADDRESSING: serve per indirizzare direttamente tutti e 64 registri del Scratchpad Register.

7) INDIRECT MEMORY ADDRESSING: le istruzioni che si riferiscono a questo metodo di indirizzamento implicano che il Data Counter punti il byte desiderato nella memoria.

8) RELATIVE ADDRESSING: tutte le istruzioni Branch dell'F8 usano questo sistema di indirizzamento. Quando si fa un Branch, il Program Counter è aggiornato da un indirizzo relativo contenuto nel secondo byte dell'istruzione.

Vengono ora descritte le istruzioni,una per una,in ordine alfabetico.Quando è definito il formato di una istruzione le parti opzionali sono racchiuse tra parentesi.Per es. una istruzione di questo tipo (Etichetta) ADC indica che l'istruzione ADC può avere o no l'etichetta.I termini e le abbreviazioni usate in seguito vengono riportate nelle 2 tabelle seguenti .

Tabella dei simboli per l'operando

Nval3 -	Questo simbolo è usato per indicare l'operando di una istruzione che definisce i 3 bit di ordine basso del codice oggetto dell'istruzione.
Nval4 -	Con questo simbolo si indica l'operando di una istruzione che definisce i 4 bit di ordine basso del codice oggetto.
Nval8 -	Questo simbolo indica l'operando di una istruzione che definisce gli 8 bit del secondo byte del codice oggetto.
Nval16-	Questo simbolo è usato per indicare l'operando di una istruzione che definisce gli 8bit del secondo byte,più gli 8 bit del terzo byte del codice oggetto dell'istruzione.

Tabella che indica il modo di riferimento ai registri

Valore o Simbolo per Sreg	Registro Scratchpad
da 0 a 11	I primi 12 registri Scr. sono indirizzati direttamente.
12 o S	L'indirizzo del registro Scr. è stabilito indirettamente dall'ISAR.
13 o I	Come per il 12,ma i 3 bit di ordine basso dell'ISAR sono incrementati dopo che si è acceduto al registro Scr..
14 o D	Come per il 12,ma i 3 bit di ordine basso dell'ISAR sono decrementati dopo che si è acceduto al registro Scr. .

ADC - ADD ACCUMULATOR TO DATA COUNTER (Aggiungi il contenuto dell'accumulatore al data counter DCØ)

Il contenuto dell'accumulatore è considerato come numero binario con segno ed è aggiunto al contenuto del registro DCO. Il risultato è immagazzinato nel registro DCO. Il contenuto dell'accumulatore non è cambiato.

Formato:

(Etichetta) ADC

Condizioni di stato:

i bit di stato non sono modificati.

Esempi:

1) Supponiamo che il contenuto dell'accumulatore sia H'3E' e che il registro DCO contenga H'209A'. Dopo l'esecuzione dell'istruzione ADC, il registro DCO conterrà H'20D8'.

```
 209A
   3E
H'20D8
```

2) Supponiamo che l'accumulatore contenga H'A2' e il registro DCO contiene H'213E'. Nella notazione in complemento a 2, H'A2' è un numero negativo, poichè il bit di ordine più alto del byte è 1.

H'A2' = 10100010
↑
Bit di segno = 1 → numero negativo

In conformità a quanto detto sopra dopo l'esecuzione dell'istruzione ADC, il registro DCO conterrà H'20E0'.

213E
FFA2
――――
H'20E0'

AI - ADD IMMEDIATE TO ACCUMULATOR (Somma diretta al contenuto dell'accumulatore)

Il numero di otto bit (due cifre esadecimali) stabilito dall'operando dell'istruzione è sommato al contenuto attuale dell'accumulatore. Viene eseguita l'addizione binaria.

Formato:
(Etichetta) AI Nval8

Condizioni di stato:
Sono modificati i bit di stato OVF, ZERO, CARRY, SIGN.
Non è modificato il bit di stato ICB.

Esempio:

Supponiamo che l'accumulatore contenga H'3F'. Dopo l'esecuzione dell'istruzione:

AI H'7E'

l'accumulatore conterrà H'BD'

H'3F' = 00111111
H'7E' = 01111110
H'BD' = 10111101

Non c'è riporto dal bit 7, così CARRY = 0

C'è riporto dal bit 6 e non dal bit 7, allora OVF = 0 ⊕ 1 = 1

Il risultato non è zero, così ZERO = 0

Il bit di ordine più alto del risultato è 1, allora SIGN = 0

AM - ADD (BINARY) MEMORY TO ACCUMULATOR (Somma (Binaria) del contenuto di una locazione di memoria con il contenuto dell'accumulatore)

Il contenuto della locazione di memoria indirizzata dal registro DCO è aggiunto al contenuto dell'accumulatore. Il risultato della somma è posto nell'accumulatore, la memoria non è alterata. Viene fatta un'addizione binaria. Il contenuto del registro DCO è incrementato di 1.

Formato:
(Etichetta) AM

Condizioni di stato:
Sono modificati i bit di stato: OVF, ZERO, SIGN, CARRY.
Inalterato ICB.

Esempio:

Supponiamo che l'accumulatore contenga H'C2',il registro DCO contenga H'213E' e la locazione di memoria H'213E' contenga H'2A'.Dopo che l'istruzione AM è stata eseguita,il registro DCO conterrà H'213F',e l'accumulatore conterrà H'EC':

H'C2' = 11000010
H'2A' = 00101010
H'EC' = 11101100

Non c'è riporto dal bit 7,allora CARRY = 0.

Non c'è riporto dal bit 6,nè dal bit 7,così OVF = $0 \oplus 0 = 0$.

Il risultato non è zero,così ZERO = 0.

Il bit di ordine più alto del risultato è 1,allora SIGN = 0.

AMD - DECIMAL ADD,MEMORY TO ACCUMULATOR (Addizione decimale Memoria - Accumulatore)

L'accumulatore e la locazione di memoria indirizzata dal registro DCO vengono considerati come contenenti due cifre BCD. Il contenuto della locazione di memoria indirizzata dal DCO è aggiunto al contenuto dell'accumulatore ottenendo un risultato BCD nell'accumulatore.

Addizione decimale : un'addizione decimale è effettuata eseguendo una addizione binaria di H'66' con uno dei due numeri BCD e quindi eseguendo l'istruzione AMD.

Formato:

AI H'66' (Precede sempre AMD per l'addizione)

(Etichetta) AMD

Sottrazione decimale : per effettuare la sottrazione decimale si deve complementare uno degli operandi e quindi eseguire la istruzione AMD.

COM (complementa l'accumulatore)

AMD

Condizioni di stato :

i bit di stato modificati sono : CARRY, ZERO

non modificato : ICB

bit di stato non significativi : OVF , SIGN

Esempi :

1)Si considera che l'accumulatore contenga H'37',il registro DCO contiene H'12FA' e la locazione di memoria H'12FA' contiene H'60'.Dopo l'esecuzione di :

AI H'66'

AMD

l'accumulatore conterrà H'97' e il registro DCO conterrà H'12FB'.

Non c'é riporto dal bit 7,allora CARRY = 0.

Altri indicatori di stato sono modificati ma questa condizione

non è significativa.

2) Si considera che l'accumulatore contenga H'79',il registro DCO contiene H'32A7',la locazione di memoria H'32A7' contiene H'80' e il registro 0 dello Scratchpad contiene H'01'.

Dopo l'esecuzione di :

COM

AMD

l'accumulatore contiene H'01'.

Non c'è riporto dal bit 7,quindi CARRY = 0
Per gli altri indicatori di stato vale il discorso dell'esempio 1).

AS - BINARY ADDITION,SCRATCHPAD MEMORY TO ACCUMULATOR (Addizione binaria,Registri - Accumulatore)

Il contenuto del registro assegnato dall'operando dell'istru - zione (Sreg) è aggiunto all'accumulatore usando l'addizione binaria.Il risultato dell'addizione binaria è immagazzinato nel l'accumulatore.Il contenuto del registro rimane invariato.A seconda del valore del Sreg,l'ISAR può essere inalterato,in - crementato o decrementato.

Formato :

(Etichetta) AS Sreg

Condizioni di stato :

Tutti i bit di stato,escluso ICB,sono modificati.

Esempio :

Supponiamo che l'accumulatore contenga H'34' e registro 11 contenga H'72'. Dopo che l'istruzione :

AS 11

è eseguita,l'accumulatore conterrà H'A6' :

```
H'34' = 00110100
H'72' = 01110010
        --------
        10100110
```

Non c'è riporto dal bit 7, così CARRY = 0

C'è riporto dal bit 6, ma non dal bit 7, così OVF = 0 ⊕ 1 = 1

Il risultato non è 0, così ZERO = 0

Il bit di ordine più alto del risultato è 1, così SIGN = 0

Supponiamo che l'accumulatore contenga H'7E', l'ISAR contenga O'27', e il registro 23 (=O'27') contenga H'A2'. Dopo che l'istruzione :

AS 13

è stata eseguita, l'accumulatore conterrà H'20', e l'ISAR sarà incrementato (solamente la cifra ottale di ordine più basso) a O'20'.

```
H'7E' = 01111110
H'A2' = 10100010
        --------
        00100000
```

C'è riporto dal bit 7, così CARRY = 1

C'è riporto dal bit 6 e dal bit 7, così OVF = 1 ⊕ 1 = 0

Il risultato è diverso da 0, così ZERO = 0

Il bit di ordine più alto del risultato è 0, così SIGN = 1

Se l'operando dell'istruzione AS fosse stato 14, il contenuto dell'ISAR, sarebbe stato decrementato a O'26', se l'operando dell'istruzione AS fosse stato 12, il contenuto dell'ISAR sarebbe rimasto invariato.

<u>ASD - DECIMAL ADD, SCRATCHPAD TO ACCUMULATOR (Addizione decimale Registro - Accumulatore)</u>

L'istruzione ASD è simile all'istruzione AMD, eccetto che invece di sommare il contenuto del byte di memoria indirizzato dal registro DCO, il contenuto del registro indirizzato dall'operando (Sreg), è sommato all'accumulatore.

Formato : Addizione decimale

AI H'66' (Precede sempre ASD per l'addizione)

(Etichetta) ASD Sreg

Formato : Sottrazione decimale

COM

(Etichetta) ASD Sreg (Precede sempre ASD per la sottrazione)

Condizioni di stato :

per i bit di stato valgono le stesse considerazioni fatte per l'istruzione AMD.

Esempio : Addizione decimale.

Si considera che l'accumulatore contenga H'42',l'ISAR contenga O'54',e il registro O'54' contenga H'83'.

Dopo che è stata eseguita la sequenza di istruzioni :

AI H'66'

ASD 14

l'accumulatore conterrà H'25'.L'ISAR conterrà O'53'.

Non c'é riporto dal bit 7,perciò CARRY = 1.

Gli altri indicatori di stato sono modificati,ma questa condizione non è significativa.

<u>BRANCH INSTRUCTION (Istruzioni di "branch")</u>

L'istruzione di branch è usata per modificare la sequenza di eseeuzione delle istruzioni del programma cambiando il contenuto del Program Counters PCO.In una istruzione di branch condizionato,il cambiamento avviene quando una certa specificata condizione è verificata,in una istruzione di branch incondizionato,il "salto" avviene semplicemente come risultato della

esecuzione della istruzione.

Tutte le istruzioni di branch sono istruzioni di due byte. Il primo byte è il codice oggetto dell'istruzione. Il secondo byte è un numero binario con segno che esprime l'entità del salto da eseguire. Poichè l'operando è di 8 bit si può coprire un campo di + 127 + - 128 locazioni rispetto alla istruzione di branch.

Il codice operativo di una istruzione di branch condizionato può essere : BC, BF, BM, BNC, BNO, BNZ, BP, BR7, BT, BZ

Per un branch incondizionato : BR

Formato :

(Etichetta) OP DEST

OP rappresenta il codice operativo che può essere BC, BM, BNC, BNO, BNZ, BP, BR7, oppure BZ.

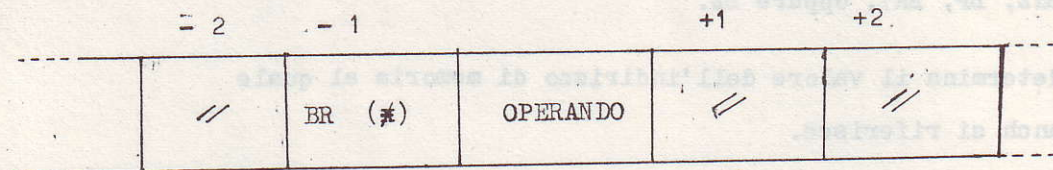
DEST determina il valore dell'indirizzo di memoria al quale un branch si riferisce.

Se il codice operativo è BF o BT, l'operando diventa t,DEST dove t può assumere valori da 0 a F a seconda di certe condizioni specificate nella tabella riportata a pag. 3.19 .

ESEMPIO DI BRANCH

INDIRIZZO	MACHINE CODE	LABEL	MNEMONIC OP CODE	OPERAND		FUNCTION
ØØØØ	71	LOOP	LIS	H'Ø1'	;	carica 01 in Acc.
ØØØ1	C2		AS	2	;	aggiungi R2 all'Acc.
ØØØ2	52		LR	2,A	;	metti il ris.in R2
ØØØ3	25 16		CI	H'16'	;	confronta Acc. con H'16'
ØØØ5	94 FA (D'-6')		BNZ	LOOP	;	se non sono uguali torna alla LOOP (6 bytes indietro)

COME SI CONTANO I BYTES PER IL BRANCH :

-2	-1		+1	+2
//	BR (≠)	OPERANDO	//	//

Ogni casella rappresenta una locazione di memoria.

(≠) oppure : BC, BF, BM, BNC ....

OPERAND t	STATUS FLAGS TESTED ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	Do not branch	An effective 3 cycle NO-OP
1	0	0	1	Branch if Positive	Same as BP
2	0	1	0	Branch on Carry	Same as BC
3	0	1	1	Branch if Positive or on Carry	
4	1	0	0	Branch if Zero	Same as BZ
5	1	0	1	Branch if Positive	Same as t=1
6	1	1	0	Branch if Zero or on Carry	
7	1	1	1	Branch if Positive or or on Carry	Same as t=3

Tabella A - Condizioni di Branch per l'istruzione BT

INSTRUCTION MNEMONIC	BRANCH WILL OCCUR IF
BC - BRANCH ON CARRY	Carry bit is set
BF - BRANCH ON FALSE	
BM - BRANCH ON NEGATIVE	Sign bit is reset
BNC - BRANCH IF NO CARRY	Carry bit is reset
BNO - BRANCH IF NO OVERFLOW	OVF bit is reset
BNZ - BRANCH IF NOT ZERO	Zero bit is reset
BP - BRANCH IF POSITIVE	Sign bit is set
BR - UNCONDITIONAL BRANCH	Always
BR7 - BRANCH ON ISAR	Any of the low 3 bits of ISAR are reset
BT - BRANCH ON TRUE	
BZ - BRANCH ON ZERO	Zero bit is set

Tabella C - Condizioni di Branch

OPERAND t	STATUS FLAGS TESTED OVF	ZERO	CARRY	SIGN	DEFINITION	COMMENTS
0	0	0	0	0	Unconditional Branch relative	
1	0	0	0	1	Branch on negative	Same as BM
2	0	0	1	0	Branch if no carry	Same as BNC
3	0	0	1	1	Branch if no carry and negative	
4	0	1	0	0	Branch if not zero	Same as BNZ
5	0	1	0	1		Same as t=1
6	0	1	1	0	Branch if no carry and result is no zero	
7	0	1	1	1		Same as t=3
8	1	0	0	0	Branch if there is no overflow	Same as BNO
9	1	0	0	1	Branch if negative and no overflow	
A	1	0	1	0	Branch if no overflow and no carry	
B	1	0	1	1	Branch if no overflow, no carry & negative	
C	1	1	0	0	Branch if no overflow and not zero	
D	1	1	0	1		Same as t=9
E	1	1	1	0	Branch if no overflow, no carry & not zero	
F	1	1	1	1		Same as t=B

Tabella B - Condizioni di Branch per l'istruzione BF

CI - COMPARE IMMEDIATE (Confronto immediato)

Il contenuto dell'accumulatore è sottratto dall'operando dell'istruzione CI.Il risultato non è salvato ma i bit di stato sono messi a 1 o a 0 per rispecchiare il risultato dell'operazione.L'accumulatore non viene alterato.

Formato :

(Etichetta) CI Nval8

Il significato Nval8 è già stato detto precedentemente.

Condizioni di stato :

Tutti i bit di stato sono modificati,escluso ICB

Esempio :

Consideriamo che l'accumulatore contenga H'1B' e il secondo byte dell'istruzione contenga H'D8'.Il confronto è fatto come segue

```
             H'1B'    00011011
complemento a 2       11100101
             H'D8'    11011000
             H'BO' 1  10111101
```

Il risultato H'BO' non è salvato.C'è riporto sul bit 7,perciò CARRY = 1.

C'è riporto sul bit 6,così OVF = 1 ⊕ 1 = 0.

Il risultato non è 0,allora ZERO = 0.

Il bit di ordine più alto è 1,allora SIGN = 0.

CLR - CLEAR ACCUMULATOR (Azzera l'accumulatore)

Il contenuto dell'accumulatore è posto a zero.

Formato :

(Etichetta) CLR

Condizioni di stato :

I bit di stato non sono modificati.

Esempio:

Consideriamo che l'accumulatore contenga H'AO'.Dopo che è stata eseguita l'istruzione CLR,l'accumulatore contiene H'00'.

CM - COMPARE MEMORY TO ACCUMULATOR (Paragona il contenuto della memoria con quello dell'Accumulatore)

L'istruzione CM è la stessa che l'istruzione CI eccetto che il paragone avviene tra il contenuto della memoria indirizzato dal registro DCO e il contenuto dell'accumulatore.

Il contenuto della memoria e quello dell'accumulatore non sono modificati.Il contenuto del registro DCO è INCREMENTATO (di una unità)

Formato :

(Etichetta) CM

COM - COMPLEMENT (Complemento (a 1))

L'accumulatore è caricato con il suo complemento a 1.

Formato :

(Etichetta) COM

Condizioni di stato :

I bit di stato modificati sono ZERO e SIGN.

Quelli posti a 0 sono OVF e CARRY.

Inalterato rimane ICB.

Esempio :

Se l'accumulatore contiene H'8B',dopo che l'istruzione COM è stata eseguita,l'accumulatore conterrà H'74'.

Il bit di ZERO è posto a 0 poichè il risultato non è zero.

Il bit SIGN è posto a 1 poichè il bit di ordine più alto del risultato è 0.

I bits OVF e CARRY sono incondizionatamente posti a 0.

DCI - LOAD DC IMMEDIATE (Carica il DC direttamente)

L'istruzione DCI è una istruzione di tre byte.Il contenuto del secondo byte sostituisce il byte di ordine alto del registro DCO,il contenuto del terzo byte sostituisce il byte di ordine

basso del registro DCO.

Formato :

(Etichetta)   DCI   Nval16

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Dopo che l'istruzione DCI H'2317' è stata eseguita, il registro DC conterrà H'2317'.

DI - DISABLE INTERRUPT (Disabilita le interruzioni)

Il bit di controllo delle interruzioni,ICB,è posto a zero,nel qual caso una richiesta di interruzione non sarà riconosciuta dalla CPU.

Formato :

(Etichetta)   DI

Condizioni di stato :

Il bit ICB è posto a 0,gli altri rimangono invariati.

DS - DECREMENT SCRATCHPAD CONTENT (Decrementa il contenuto dello Scratchpad)

Il contenuto del registro dello Scratchpad indirizzato dallo operando (Sreg) è decrementato di una unità.L'operazione di decrementazione è eseguita aggiungendo H'FF' (D'-1') al re - gistro specificato.

Formato :

(Etichetta) DS Sreg

Condizioni di stato :

Vengono modificati tutti i bit di stato, escluso ICB.

Esempio :

Consideriamo che l'ISAR contenga O'23' e il registro O'23' contenga H'17'.Dopo che è stata eseguita l'istruzione DS D, il registro O'23' contiene H'16' e l'ISAR contiene O'22'.

C'è riporto sul bit 7,perciò CARRY = 1.
C'è riporto sul bit 6,allora OVF = 1 $\oplus$ 1 = 0.
Il risultato dell'operazione di decrementazione non è 0, allora ZERO = 0.
Il bit più significativo è 0,allora SIGN = 1.

EI - ENABLE INTERRUPT (Abilita le interruzioni)

Il bit di controllo delle interruzioni è posto a 1.Una richiesta di interruzione sarà ora riconosciuta dal CPU.

Formato :

(Etichetta)   EI

Condizioni di stato :

ICB a 0, gli altri bit di stato non sono modificati.

IN - INPUT LONG ADDRESS (Ingresso)

Il dato in ingresso dalla porta I/O specificata dall'operando dell'istruzione IN è immagazzinato nell'accumulatore.

Le porte I/O con indirizzi da 4 a 255 possono essere indirizzate dall'istruzione IN.Le porte I/O con indirizzo da 0 a 15 possono essere accessibili dall'istruzione INS.Le porte 0 - 3 DEVONO essere indirizzate con l'istruzione INS.
L'istruzione IN si rappresenta in due byte del codice oggetto, mentre l' istruzione INS in un byte del codice oggetto.
IMPORTANTE : se una porta I/O è stata usata sia per ingresso che per uscita,una porta precedentemente usata per uscita deve essere azzerata prima che possa essere usata per ingresso di dati.

Formato :

(Etichetta)   IN   Nval8

Condizioni sullo stato :

I bit di stato modificati sono : ZERO, SIGN

Azzerati : OVF, CARRY

Inalterato : ICB

Esempio :

Assumiamo che il valore H'37' sia stato mandato in ingresso da un'unità esterna alla porta I/O H'10'.Dopo che è stata eseguita l'istruzione IN H'10',l'accumulatore conterrà H'37'.

OVF e CARRY sono incondizionatamente messi a 0.Il contenuto dell'accumulatore non è zero,perciò ZERO = 0,il bit più significativo è 0 perciò SIGN = 1.

INC - INCREMENT ACCUMULATOR (Incrementa l'accumulatore)

Il contenuto dell'accumulatore è incrementato di un numero binario.

Formato :

(Etichetta) INC

Condizioni sullo stato :

Tutti i bit di stato sono modificati tranne ICB.

Esempio :

Consideriamo che l'accumulatore contenga H'FF'.Dopo la esecuzione di un'istruzione INC,l'accumulatore contiene H'00'.

CARRY = 1

OVF = $1 \oplus 1 = 0$

ZERO = 1 e SIGN = 1

INS - INPUT SHORT ADDRESS (Ingresso formato corto)

Il dato in ingresso alla porta I/O specificato dall'operando dell'istruzione INS è caricato nell'accumulatore.Una porta I/O con un indirizzo che comprende un intervallo da 0 a 15 può essere resa accessibile da questa istruzione.Le porte 0 - 3 DEVONO essere indirizzate con l'istruzione INS.
IMPORTANTE : come per IN,se una porta I/O è stata usata sia come ingresso sia come uscita,la porta precedentemente usata per uscita deve essere azzerata prima ~~prima~~ che possa essere usata come ingresso dati.

Formato :

(Etichetta)   INS   NVal4

Condizioni sullo stato :

ZERO e SIGN modificati

OVF, CARRY posti a zero

ICB non modificato

Esempio :

Consideriamo che la porta I/O della CPU indirizzata da H'01' contenga H'86'.L'esecuzione dell'istruzione INS 1

fa si che l'accumulatore sia caricato con H'86'.

OVF = CARRY = 0

ZERO = 0
SIGN = 0.

JMP - BRANCH IMMEDIATE (Salto diretto)

Come risultato dell'esecuzione di un'istruzione di JMP, si ha un "salto" alla locazione di memoria indirizzata dal secondo e dal terzo byte dell'istruzione. Il secondo byte contiene gli otto bit di ordine alto dell'indirizzo di memoria: il terzo byte contiene gli otto bit di ordine basso dell'indirizzo di memoria.

L'accumulatore è usato per immagazzinare temporaneamente il byte più significativo dell'indirizzo di memoria; perciò, dopo che è stata eseguita un'istruzione di JMP, IL CONTENUTO INIZIALE DELL'ACCUMULATORE E' PERSO.

Formato :

(Etichetta) JMP Nval16

Condizioni di stato :

I bit di stato non sono alterati.

Esempio :

Consideriamo che l'operando dell'istruzione JMP contenga H'03A4'.Dopo che l'istruzione JMP H'03A4' è stata eseguita avviene un "salto" all'indirizzo H'03A4'. A completamento di questa esecuzione l'accumulatore contiene H'03'.

LI - LOAD IMMEDIATE (Carico immediato)

Il valore stabilito dall'operando dell'istruzione LI è caricato nell'accumulatore.

Formato :
(Etichetta) LI Nval8

Condizioni di stato :
I bit di stato non sono alterati.

Esempio :

Consideriamo che il secondo byte dell'istruzione LI contenga H'C7'. L'istruzione LI H'C7' fa si che l'accumulatore sia caricato con H'C7'.

LIS - LOAD IMMEDIATE SHORT (Carico immediato di tipo corto)

Un valore di 4 bit stabilito dall'operando dell'istruzione LIS

è caricato nei 4 bit meno significativi dell'accumulatore.
I 4 bit più significativi sono posti a 0.

Formato :
(Etichetta) LIS Nval4

Condizioni di stato :
I bits di stato non sono modificati.

Esempio :

Dopo che l'istruzione LIS 3 è stata eseguita,l'accumulatore conterrà H'03'.

LISL - LOAD LOWER OCTAL DIGIT OF ISAR (Carica la cifra ottale più bassa dell'ISAR)

Un valore di 3 bit stabilito dall'operando dell'istruzione LISl è caricato nei 3 bit meno significativi dell'ISAR.I 3 bit più significativi dell'ISAR non sono modificati.

Formato :
(Etichetta) LISL Nval3

Condizioni di stato :
I bits di stato non sono modificati.

Esempio :

Supponiamo che l'ISAR contenga il valore O'72'.Dopo che l'istruzione LISL 7 è stata eseguita,l'ISAR conterrà il valore O'77'.

LISU - LOAD UPPER OCTAL DIGIT OF ISAR (Carica la cifra ottale più alta dell'ISAR)

Un valore di 3 bit,stabilito dall'operando dell'istruzione LISU,è caricato nei 3 bit più significativi dell'ISAR. I 3 bit meno significativi dell'ISAR non sono alterati.

Formato :

(Etichetta) LISU Nval3

Condizioni di stato :
I bit di stato non sono cambiati.

Esempio :

Supponiamo che l'ISAR contenga il valore O'72'.Dopo che è stata eseguita l'istruzione LISU 2, l'ISAR conterrà il valore O'22'.

LM - LOAD ACCUMULATOR FROM MEMORY (Carica l'accumulatore dalla memoria)

Il contenuto del byte di memoria indirizzato dal registro DCO è caricato nell'accumulatore.Il contenuto del registro DCO è incrementato come un risultato dell'esecuzione dell'istruzione LM.

Formato :

(Etichetta) LM

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che il registro DCO contenga H'37A2' e la locazione di memoria indirizzata da H'37A2' contenga H'2B'.L'esecuzione dell'istruzione LM fa si che l'accumulatore sia caricato con H'2B'.Il registro DCO successivamente conterrà H'37A3'.

LNK - LINK CARRY TO THE ACCUMULATOR (Unisci il riporto all'accumulatore)

Il bit di riporto è sommato in modo binario al bit meno significativo dell'accumulatore.Il risultato è immagazzinato nello accumulatore.

Formato :

(Etichetta) LNK

Condizioni di stato :

I bit di stato modificati sono : OVF , ZERO , CARRY , SIGN.

ICB non è modificato.

Esempio :

Consideriamo che l'accumulatore contenga H'84' e il bit CARRY sia posto a 1.L'esecuzione dell'istruzione fa si che l'accumulatore contenga H'85.Nel risultato dell'esecuzione dell'istruzione non c'è riporto sul bit 7,perciò CARRY = 0. Non c'è neppure riporto sul bit 6,perciò OVF = 0 ⊕ 0 = 0.
Il risultato è diverso da zero,quindi ZERO = 0.
Il bit più significativo del risultato è 1,perciò SIGN = 0.

LR - LOAD REGISTER (Carica il registro)

Il gruppo di istruzioni LR muove uno o due bytes di dati fra un registro "sorgente" ed un registro "destinazione".Esistono istruzioni che muovono i dati fra i seguenti registri :

a) Un registro Scratchpad e l'accumulatore
b) Un registro " e il Data Counter (DCO)
c) L'accumulatore e l'ISAR
d) Il registro Scratchpad 9 e il registro di stato
e) I registri " e il Program Counter (PCO)
f) I registri " e lo Stack Register (PC1)

Una sorgente e destinazione di dati di una istruzione LR è determinata dall'operando dell'istruzione. Il numero di dati mossi (uno o due) dipende dalla lunghezza della sorgente e dei registri di destinazione (8 o 16 bits)

Formato :

(Etichetta) LR D,S

S è il registro sorgente

D è il registro destinazione

Condizioni di stato :

I bits di stato non sono modificati.

Esempio :

Consideriamo che l'ISAR contenga O'76'. Dopo che l'istruzione LR A,IS è stata eseguita, l'accumulatore contiene O'76'. Il registro Scratchpad O'76' rimane invariato.

Tabella D . Definizione dell'operando dell'istruzione LR.

LR INSTRUCTION OPERANDS DESTINATION	SOURCE	LOADS REGISTER	FROM REGISTER	WITH
A,	KU	Accumulator	Scratchpad register 12	8-bit contents
A,	KL	Accumulator	Scratchpad register 13	8-bit contents
A,	QU	Accumulator	Scratchpad register 14	8-bit contents
A,	QL	Accumulator	Scratchpad register 15	8-bit contents
KU,	A	Scratchpad register 12	Accumulator	8-bit contents
KL,	A	Scratchpad register 13	Accumulator	8-bit contents
QU,	A	Scratchpad register 14	Accumulator	8-bit contents
QL,	A	Scratchpad register 15	Accumulator	8-bit contents
K,	P	Scratchpad register 12	Program Counter PC1	High order 8-bit byte
		Scratchpad register 13	Program Counter PC1	Low order 8-bit byte
P,	K	High order byte of PC1	Scratchpad register 12	8-bit contents
		Low order byte of PC1	Scratchpad register 13	8-bit contents

NI - AND IMMEDIATE (Immediato)

Su di un valore di 8 bit stabilito dall'operando dell'istruzione NI è eseguita l'operazione AND con il contenuto dello accumulatore. Il risultato è immagazzinato nell'accumulatore.

Formato :

(Etichetta) NI Nval8

Condizioni di stato :

I bit OVF e CARRY sono posti a zero, vengono variati ZERO e SIGN, e non si ha nessun effetto su ICB.

Esempio :

Consideriamo che il secondo byte dell'istruzione NI contenga H'36' e l'accumulatore contenga H'2A', come risultato dell'esecuzione dell'istruzione l'accumulatore contiene H'22'.

H'36' = 00110110
H'2A' = 00101010
H'22' = 00100010

Non c'è riporto sul bit 7, così CARRY = 0

Non c'è riporto sul bit 6, allora OVF = 0 $\oplus$ 0 = 0

Il risultato è diverso da 0, così ZERO = 0

Il bit più significativo è zero, perciò SIGN = 1

NM - LOGICAL AND FROM MEMORY (AND con la memoria)

Sul contenuto della memoria indirizzato dal registro DCO è fatta l'operazione di AND con il contenuto dell'accumulatore. I risultati sono immagazzinati nell'accumulatore. Il contenuto del registro DCO è incrementato.

Formato :
(Etichetta) NM

Condizioni di stato :
OVF e CARRY sono posti a 0, vengono modificati i bit ZERO e SIGN, ICB resta invariato.

Esempio :

Consideriamo che il registro DCO contenga H'49AC', la locazione di memoria indirizzata da H'49AC' contenga H'67' e l'accumulatore contenga H'A9'. Dopo l'esecuzione dell'istruzione NM, l'accumulatore contiene H'21', e il Data Counter contiene H'49AD'.

H'67' = 01100111
H'A9' = 10101001
H'21' = 00100001

NOP - NO OPERATION (Nessuna operazione)

Non viene eseguita nessuna funzione ma solo incrementato il PCO.

Formato :

(Etichetta) NOP

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che il PCO contenga H'2700'.Dopo che l'istruzione NOP è stata eseguita,il registro PCO contiene H'2701'.

NS - LOGICAL AND FROM SCRATCHPAD MEMORY (AND registro)

Sul contenuto del registro indirizzato dall'operando (Sreg) viene eseguita l'operazione AND con il contenuto dell'accumulatore.Il risultato è immagazzinato nell'accumulatore.

Formato :
(Etichetta) NS Sreg

Condizioni di stato :

OVF e CARRY sono posti a 0, ZERO e SIGN sono modificati e su ICB non si ha nessun effetto.

Esempio :

Consideriamo che un registro contenga H'F2' e l'accumulatore contenga H'2F'.L'esecuzione dell'istruzione NS 2 fa si che l'accumulatore contenga H'22.

```
H'F2' = 11110010
H'2F' = 00101111
H'22' = 00100010
```

Non c'è riporto sul bit 7, perciò CARRY = 0 e non c'è neppure riporto sul bit 6, quindi OVF = 0 ⊕ 0 = 0. Il risultato non è 0, così ZERO = 0. Il bit più significativo del risultato è 0, perciò SIGN = 1.

OI - OR IMMEDIATE (OR immediato)

Su di un valore di 8 bit, stabilito dall'operando dell'istruzione OI, è eseguita l'operazione OR con il contenuto dell'accumulatore. Il risultato è immagazzinato nell'accumulatore.

Formato :

(Etichetta)   OI   Nval8

Condizioni di stato :

ZERO e SIGN sono modificati, OVF e CARRY sono posti a 0 e ICB rimane invariato.

Esempio :

Consideriamo che l'accumulatore contenga H'OA'. L'esecuzione dell'istruzione OI H'A3' fa si che l'accumulatore contenga H'AB'.

```
H'A3' = 10100011
H'0A' = 00001010
H'AB' = 10101011
```

Il risultato nell'accumulatore non è 0,perciò ZERO = 0.

Il bit più significativo del risultato è 1,quindi SIGN = 0.

OVF e CARRY sono posti a 0.

OM - LOGICAL OR FROM MEMORY (OR con la memoria)

Sul contenuto del byte di memoria,indirizzato dal rigistro DCO, è eseguita l'operazione OR con il contenuto dell'accumulatore. Il risultato è immagazzinato nell'accumulatore.Il registro DCO è incrementato.

Formato :

(Etichetta) OM

Condizioni di stato :

Modificati ZERO e SIGN, posti a 0 OVF e CARRY, invariato ICB.

Esempio :

Consideriamo che il registro DCO contenga H'FC19',la locazione di memoria indirizzata da H'FC19' contenga H'16' e l'accumulatore contenga H'81'.Dopo l'esecuzione dell'istruzione OM lo accumulatore conterrà H'97' e il registro DCO conterrà H'FC1A'.

```
H'16' = 00010110
H'81  = 10000001
H'97' = 10010111
```

Il risultato non è zero,perciò ZERO = 0

Il bit più significativo del risultato è 1, così SIGN = 0

I bit di overflow e di carry sono incondizionatamente posti a 0, così OVF = 0 e CARRY = 0.

OUT - OUTPUT LONG ADDRESS (Uscita)

La porta I/O indirizzata dall'operando dell'istruzione OUT è caricata con il contenuto dell'accumulatore.

Le porte I/O con indirizzi da 4 a 255 possono essere rese accessibili con l'istruzione OUT (per le 1 - 4 si deve usare l'istruzione OUTS).

L'istruzione OUT genera due bytes del codice oggetto,mentre l'istruzione OUTS genera un byte del codice oggetto.

Formato :

(Etichetta) OUT Nval8

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che l'accumulatore contenga H'2A'.L'esecuzione OUT H'F6' farà si che la porta I/O H'F6' sia caricata con H'2A'.

OUTS - OUTPUT SHORT ADDRESS (Uscita formato corto)

La porta I/O indirizzata dall'operando dell'istruzione OUTS è caricata con il contenuto dell'accumulatore.Le porte I/O con indirizzi da 0 a 15 possono essere rese accessibili da questa istruzione.

Formato :

(Etichetta)   OUTS Nval4

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :   Consideriamo che l'operando (Nval4) dell'istruzione OUTS, sia H'OF', e l'accumulatore contenga H'32' . L'esecuzione dell'istruzione OUTS 15 farà si che la porta I/O H'OF' contenga H'32'.

PI - PROGRAM COUNTER IMMEDIATE (Chiamata ad un sottoprogramma)

Il contenuto del Program Counter è immagazzinato nello Stack Register ,PC1, allra l'indirizzo a 16 bit contenuto nell'operando dell'istruzione PI è caricato nel Program Counter .L'accumulatore è usato come registro di temporaneo immagazzinamento durante il trasferimento del byte più significativo dell'indirizzo.

IL CONTENUTO PRECEDENTE DELL'ACCUMULATORE SARA' ALTERATO.

Formato :

(Etichetta) PI Nval16

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che l'operando dell'istruzione PI contenga H'32A1', il Program Counter (PCO) contenga H'ABCD' e lo Stack Register (PC1) contenga H'1234'. L'esecuzione dell'istruzione PI H'32A1' fa si che lo Stack Register contenga H'ABCD' e il Program Counter contenga H'32A1'.

PK - PROGRAM COUNTER FROM REGISTER K (Chiama ad un sottoprogramma il cui indirizzo è nel registro K)

Il contenuto del registro Program Counter (PCO) è immagazzinato nello Stack Register (PC1), allora il contenuto del registro K dello Scratchpad (Registri 12 e 13 dello Scratchpad) è trasferito nel registro Program Counter.

Formato :

(Etichetta) PK

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che il registro 12 dello Scratchpad contenga H'AB',il registro 13 dello Scratchpad contenga H'CD',e il registro DCO contenga H'1234'.L'esecuzione dell'istruzione PK fa si che PC1 contenga H'1234' e il registro PCO contenga H'ABCD'.

<u>POP - RETURN FROM SUBROUTINE (Ritorno da un sottoprogramma)</u>

Il contenuto del PC1 è trasferito nel PCO.

Formato :
(Etichetta) POP

Condizioni di stato :
I bit di stato non sono modificati.

Esempio :

Consideriamo che il PC1 contenga H'ABCD' e il PCO contenga H'1234'. Quando è stata eseguita l'istruzione POP,il PCO conterrà H'ABCD'.

<u>SL - SHIFT LEFT (Shift a sinistra)</u>

Il contenuto dell'accumulatore è shiftato a sinistra di 1 o 4 posizioni,ciò dipendendo dal valore dell'operando dell'istruzione SL.Se il valore dell'operando è 1 il contenuto dell'accumulatore

è shiftato di 4 posizioni. I 4 bit meno significativi diventa
no 0.

Formato :

(Etichetta) SL 1 o 4

Condizioni di stato :

Sono modificati i bit ZERO e SIGN. OVF e CARRY sono posti a
zero. ICB resta invariato .

Esempio :

Consideriamo che l'accumulatore contenga H'81'.L'esecuzio
ne dell'istruzione SL 1 fa si che l'accumulatore contenga H'02'.
L'esecuzione dell'istruzione SL 4 fa si che l'accumulatore conten
ga H'10'.
In entrambi gli esempi il risultato non è 0, perciò ZERO = 0.
Il bit più significativo del risultato è 0,perciò SIGN = 1.
I bit di overflow e di carry sono incondizionatamente posti a 0,
perciò OVF = 0 e CARRY = 0.

SR - SHIFT RIGHT (Shift a destra)

Il contenuto dell'accumulatore è shiftato a destra di 1 o 4 posi
zioni,dipendendo ciò dal valore dell'operando dell'istruzione SR.
Se il valore dell'operando è 1,il contenuto dell'accumulatore sarà

shiftato di una posizione. Il bit più significativo diventa 0.
Se il valore dell'operando è 4, il contenuto dell'accumulatore è shiftato di quattro posizioni . I 4 bit più significativi diventano 0.

Formato :

(Etichetta)   SR   1 o 4

Le condizioni di stato sono identiche a quelle di SL.

Esempio :

Consideriamo che l'accumulatore contenga H'81'. L'esecuzione dell'istruzione SR 1 fa si che l'accumulatore contenga H'40'. L'esecuzione dell'istruzione SR 4 fa si che l'accumulatore contenga H'08'.

In ambedue gli esempi si ha : ZERO = 0 , SIGN = 1 , OVF = 0 e CARRY = 0 .

ST - STORE TO MEMORY (Immagazzinamento in memoria)

Il contenuto dell'accumulatore è immagazzinato nella locazione di memoria indirizzata dal DCO. Il contenuto del DCO è incrementato dopo che è stata eseguita l'istruzione.

Formato :

(Etichetta)   ST

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che l'accumulatore contenga H'69', e il registro DCO contenga H'ABBE'. L'esecuzione dell'istruzione ST fa si che la locazione di memoria con indirizzo H'ABBE' contenga H'69'. DCO è incrementato e contiene H'ABBF'.

XDC - EXCHANGE DATA COUNTERS (Scambia il contenuto dei Data Counters)

L'esecuzione dell'istruzione XDC fa si che il contenuto del registro Data Counter ausiliario (DC1) sia scambiato con il contenuto del registro Data Counter (DCO). Questa istruzione è valida solo quando un chip 3852 o 3853 di interfaccia di memoria, fa parte della configurazione del sistema.

Formato :

(Etichetta) XDC

Condizioni di stato :

I bit di stato non sono modificati.

Esempio :

Consideriamo che il DCO contenga H'ABCD' e il DC1 contenga H'1234'. L'esecuzione dell'istruzione XDC fa si che il registro

DCO contenga H'1234' e il registro DC1 contenga H'ABCD'.

XI - EXCLUSIVE-OR IMMEDIATE (OR-esclusivo immediato)

Sul contenuto di 8 bit, stabilito dall'operando dell'istruzione XI, viene eseguita l'operazione OR-esclusivo con il contenuto dell'accumulatore. Il risultato è immagazzinato nell'accumulatore.

Formato :

(Etichetta) XI Nval8

Condizioni di stato :

ZERO e SIGN sono modificati, OVF e CARRY sono posti a 0, ICB resta invariato.

Esempio :

Consideriamo che l'accumulatore contenga H'AB' e l'operando dell'istruzione XI contenga H'42'. L'esecuzione dell'istruzione XI H'42' fasi che l'accumulatore contenga H'E9'.

```
H'AB' = 10101011
H'42' = 01000010
H'E9' = 11101001
```

Il risultato non è 0, così ZERO = 0. Il bit di ordine più alto del risultato è 1, quindi SIGN = 0. OVF e CARRY sono posti incondizionatamente a 0.

XM - EXCLUSIVE-OR FROM MEMORY (OR-esclusivo con la memoria)

Sul contenuto della locazione di memoria indirizzata da DCO è eseguita l'operazione OR-esclusivo con il contenuto dell'accumulatore. Il risultato è immagazzinato nell'accumulatore. Il registro DCO è incrementato.

Formato :

(Etichetta) XM

Condizioni di stato :

I bit di stato modificati sono ZERO e SIGN , mentre resta invariato ICB. OVF e CARRY sono posti a 0.

Esempio :

Consideriamo che il DCO contenga H'1DE4', la locazione di memoria indirizzata da H'1DE4' contiene H'1D', e l'accumulatore contiene H'A8'. L'esecuzione dell'istruzione XM fa si che l'accumulatore contenga H'B5'.

```
H'1D' = 00011101
H'A8' = 10101000
H'B5' = 10110101
```

Il risultato non è 0, perciò ZERO = 0 ; il bit di ordine più alto del risultato è 1, perciò SIGN = 0 ; OVF e CARRY sono posti a 0.

XS - EXCLUSIVE-OR FROM SCRATCHPAD (OR-esclusivo con un registro)

Sul contenuto del registro dello Scratchpad richiamato dall'operando (Sreg) è eseguita l'operazione OR-esclusivo con il contenuto dell'accumulatore.

Formato :

(Etichetta) XS Sreg

Condizioni di stato :

ZERO e SIGN sono modificati, OVF e CARRY sono posti a 0 e ICB resta invariato.

Esempio :

Consideriamo che il registro 10 dello Scratchpad contenga H'7B' e l'accumulatore contenga H'61'.L'esecuzione dell'istruzione XS 10 fa si che l'accumulatore contenga H'1E'.

H'7B' = 01111011
H'61' = 01100001
H'1E' = 00011110

ZERO = 0 , SIGN = 1 , OVF = 0 , CARRY = 0.

CAPITOLO 4

Esempi di programmazione

I programmi, che seguono, sono realizzati per il microcalcolatore CHILD 8 della MICROPI - Firenze -

Programma per stampare "READY"

Si vuole realizzare un programma per stampare una parola, per esempio "READY".

Realizziamo dapprima un Flow-chart:

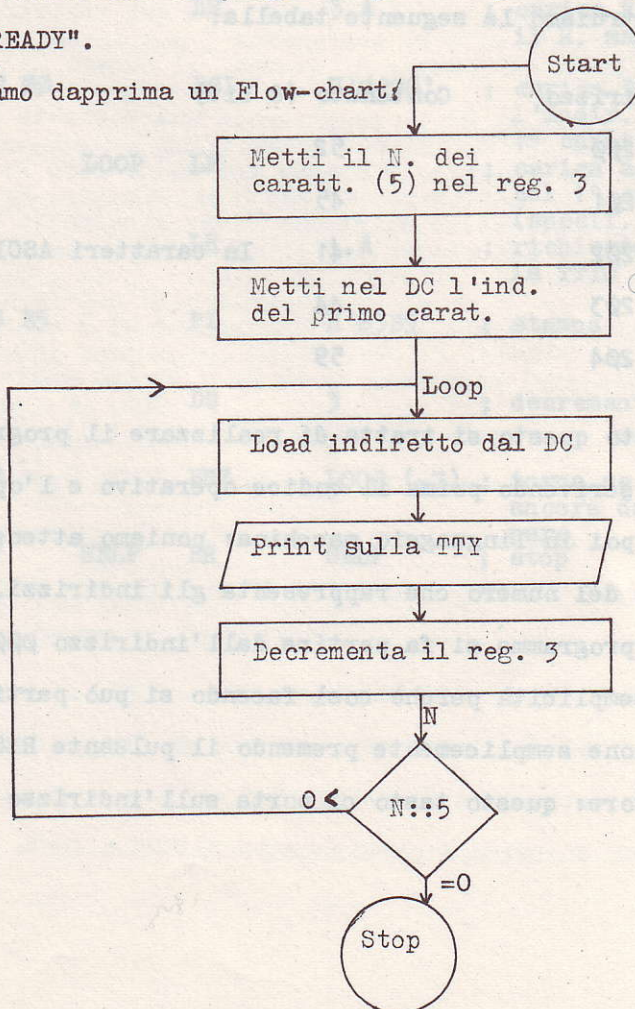



Fig. 3.3

Si può notare, nel flow-chart, che fra i 64 registri della CPU del F8, abbiamo scelto il numero 3 e vi abbiamo collocato i caratteri della parola da stampare: sono 5, essendo 5 le lettere che compongono READY.

Collochiamo in memoria, per esempio partendo dall'indirizzo 2ØØ, le lettere, che compongono READY, in codice ASCII.

Costruiamo la seguente tabella:

Indirizzo	Contenuto (8 bit)		
2ØØ	52		R
2Ø1	45		E
2Ø2	41	In caratteri ASCII	A
2Ø3	44		D
2Ø4	59		Y

Fatto questo si tratta di realizzare il programma vero e proprio scrivendo prima il codice operativo e l'operando e traducendo poi in linguaggio macchina: poniamo attenzione al cambiamento del numero che rappresenta gli indirizzi.

Il programma si fa partire dall'indirizzo ØØØØ. Questo si fa per semplicità perchè così facendo si può partire nella programmazione semplicemente premendo il pulsante RESET sul microcalcolatore: questo tasto ci porta sull'indirizzo ØØØØ.

INDIRIZZO	MACHINE CODE	LABEL	MNEMONIC OP CODE	OPERAND	FUNCTION
ØØØØ	1A		DI		; si disabilita l'interrupt
ØØØ1	2Ø Ø1		LI	H'Ø1'	; richiesto dal la TTYO
ØØØ3	5Ø		LR	Ø,A	; richiesto dal la TTYO
ØØØ4	2Ø Ø5		LI	H'Ø5'	; carica l'acc. col N. max dei caratt. (5)
ØØØ6	53		LR	3,A	; carica R3 con il N. max
ØØØ7	2A Ø2 ØØ		DCI	H'Ø2ØØ'	; carica DC col l'indir. del 1° caratt.
ØØØA	16	LOOP	LM		; carica acc. col 1° caratt. (specif. dal DC)
ØØØB	51		LR	1,A	; richiesto dal la TTYO
ØØØC	28 83 E5		PI	H'83E5'	; stampa
ØØØF	33		DS	3	; decrementa R3
ØØ1Ø	94 F9		BNZ	LOOP (-7)	; torna se c'è ancora da stam pare
ØØ12	9Ø FF	SELF	BR	SELF	; stop

Commento al programma

La prima istruzione DI è una istruzione di sicurezza e serve per azzerare il bit di controllo dell'interruzione.

Le istruzioni caratterizzate dalla scritta a lato "RICHIESTO DALLA TTYO" sono istruzioni che servono per soddisfare le condizioni necessarie sul registro 0 e sul registro 1 perchè si possa svolgere la routine di stampa. Il numero 01' che compare nell'istruzione LI H'01' rappresenta una costante che deve essere messa nel registro Ø sempre per soddisfare le sopraddette condizioni. La routine di stampa viene richiamata dall'istruzione PI H'83E5'.

Se il numero che segue l'istruzione LI è minore di 15 (e siamo nel nostro caso dove si ha 05') si può usare l'istruzione LIS che ci permette di risparmiare una cella di memoria: infatti è necessario un solo byte (vedi elenco istruzioni). Nel programma è stato eseguito un procedimento generale.

Abbiamo poi una istruzione LM che oltre a caricare l'accumulatore con il contenuto della memoria indirizzata dal DC, fa si che il DC si incrementi.

Troviamo inoltre una istruzione di Branch (sia BNZ che BR) che è una istruzione di salto. Noi abbiamo a disposizione due istruzioni di salto e precisamente BRANCH e JUMP. La prima ci

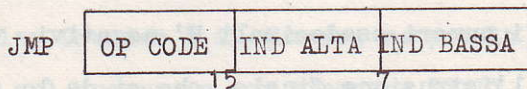
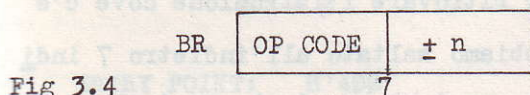
comanda un salto relativo a $\pm n$, la seconda ci comanda un salto assoluto ad un indirizzo ben determinato.

Facciamo un esempio: se da un indirizzo di memoria ØØ5 noi vogliamo tornare all'indirizzo ØØ3 noi possiamo usare il JUMP, però se per ragioni di programma l'istruzione che era all'indirizzo ØØ5 e quelle precedenti vengono trasferite all'indirizzo 1Ø5, con il JUMP non possiamo andare a 1Ø3 ma torniamo irrimediabilmente a ØØ3.

Questo non accade con l'istruzione BRANCH, che si dice relativa.

Un altro vantaggio dell'istruzione BRANCH consiste nel fatto che avendo un operando di 8 bit, si risparmia una cella di memoria rispetto al JUMP, e questo può essere molto importante nelle applicazioni del microcomputer.

Per vedere lo svantaggio analiziamo il formato dell'istruzione di BRANCH e di JUMP:

Fig 3.4

Con BR, per il salto, abbiamo a disposizione 8 bit quindi al massimo si può agire su una parte di memoria limitata a 256 indirizzi.

$2^8 = 256$ $\begin{matrix} +128 \\ -127 \end{matrix}$ Campo di memoria visualizzabile con una istruzione di BRANCH.

Mentre con JMP sono a disposizione 16 bit e quindi si può agire su tutta la memoria.

Le variazioni di indirizzo si possono comprendere sapendo che ciascun gruppo di due simboli nel "machine code" corrisponde a 8 bit e quindi comporta la variazione seguente di un indirizzo di memoria. Esempio: ØØØ7 2A Ø2 ØØ 2A è contenuto all'indirizzo ØØØ7, Ø2 all'indirizzo ØØØ8, ØØ all'indirizzo ØØØ9, e quindi il seguente è ØØØA (vedi programma).

All'istruzione di salto BNZ, che si rimanda all'etichetta LOOP, corrisponde un'istruzione in linguaggio macchina caratterizzata da un F9 (-7).

Questo vuol dire che per ritrovare l'istruzione dove c'è l'etichetta LOOP e cioè 16 abbiamo saltato all'indietro 7 indirizzi. Cioè: 94 (-1), 33 (-2), E5 (-3), 83 (-4), 28 (-5), 51 (-6), 16 (-7). Per i numeri esadecimali H' negativi, vedere l'Appendice A. Per l'istruzione finale, che ci da lo stop, il discorso è analogo.

Facciamo un salto all'indietro di -1, infatti FF è un decimale -1, e in questo modo chiudiamo il programma (anche se il calcolatore prosegue indefinitivamente a eseguire salti di -1).

E' da notare che un BR ØØ non ha significato.

Subroutine "Print"

Questa subroutine serve per provocare la stampa di un buffer di caratteri ASCII.

Si devono passare: il numero di caratteri da stampare, il loro indirizzo di partenza, il delay counter.

ENTER: R3 - Numero di caratteri
DC - Indirizzo di partenza
RØ - Delay Counter

EXIT: R3 - Ø
DC - Indirizzo di partenza + numero di carat.
RØ - Invariato
W - Distrutto
A - "
R1 - -1
R2 - Ø

ENTRY POINT: H'4ØØ'

Il Delay Counter è il numero, sempre lo stesso, che abbiamo scritto nel programma per stampare READY, nell'istruzione LI H'01', per soddisfare una delle condizioni richieste dalla TTYO. E' un numero che serve per regolare la velocità di esecuzione dell'istruzioni.

L'Exit mi rappresenta la situazione in cui si trova il micro dopo l'esecuzione della Subroutine "Print" (quelle condizioni si trovano nel manuale)

SUBROUTINE "PRINT"

```
Ø4ØØ  1A            DI              ; si disabilita
                                      l'interrupt
Ø4Ø1  Ø8            LR   K,P        ; salva il PC1
Ø4Ø2  16      LOOP  LM              ;
Ø4Ø3  51            LR   1,A        ; come visto nel
                                      programma per
Ø4Ø4  28 83E5       PI   H'83E5'      stampare "READY"
Ø4Ø7  33            DS   3          ; [PCØ]= 83E5 [PC1] = 4Ø4
Ø4Ø8  94 F9         BNZ  LOOP (-7,H'F9') ;
Ø4ØA  Ø9            LR   P,K        ; ripristina il PC1
Ø4ØB  1C            POP             ; return
```

Commento

La struttura della Subroutine " Print" è molto simile alla seconda parte del programma per stampare READY: compare sempre l'istruzione "disable interrupt" DI e in più compaiono 2 istruzioni LR K,P e LR P,K che riguardano lo stack-register.

La necessità dell'istruzione LR K,P per salvare il contenuto dello stack-register (PC1) è dovuta al fatto che all'interno della nostra Subroutine compare una Subroutine di servizio qual'è la TTYO.

Quando si richiama la Subroutine "Print" nel PC1 si porta l'indirizzo dell'istruzione successiva del programma in corso e nel PCO (Program Counter) l'indirizzo 4ØØ.

Durante l'esecuzione della TTYO il contenuto di PC1 andrebbe perduto perchè quest'ultimo si porta a 4Ø4 e nel Program Counter si immagazzina il primo indirizzo della TTYO. Per questo si manda il contenuto di PC1, quando si interrompe il programma in corso, in uno dei sessantaquattro registri della CPU, nel caso nostro, il registro K.

Possiamo fare uno schema di quello che succede nel PC1 e PCO quando si comanda PI oPOP:

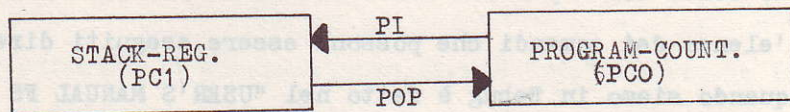



Fig. 3.5

Con PI il contenuto di PCO va in PC1 con POP viceversa.

Programma chiamante la Subroutine "Print"

Per fare questo programma chiamante ci riferiamo al programma per stampare READY. Si parte dall'indirizzo ØØØØ ricalcando la prima parte del prog. "READY" e aggiungendo due istruzioni finali.

```
ØØØØ   1A          DI              ;
ØØØ1   2Ø Ø1       LI   H'Ø1'      ; come visto nel program
                                   ; ma per stampare "READY"
ØØØ3   5Ø          LR   ØA         ; [PCØ] = ØØØ7 [PC1] = X
ØØØ4   2Ø Ø5       LI   H'Ø5'      ; [PCØ] = ØØØA [PC1] = X
ØØØ6   53          LR   3A         ; [PCØ] = Ø4ØØ [PC1] = ØØØD
ØØØ7   2A Ø2 ØØ    DCI  H'Ø2ØØ'    ;
ØØØA   28 Ø4 ØØ    PI   H'Ø4ØØ'    ; richiama la Sub. "Print"
ØØØD   28 8Ø 8Ø    PI   H'8Ø8Ø'    ;torna al Debug
```

Le ultime due istruzioni sono le modifiche al programma "READY". Con PI H'Ø4ØØ si chiama la Subroutine "Print", mentre con PI H'8Ø8Ø si torna al "Debug".

Entrare in Debug vuol dire poter eseguire certe istruzioni che sono nella ROM a partire dalla locazione di memoria H'8Ø8Ø.

L'elenco dei comandi che possono essere eseguiti direttamente quando siamo in Debug è fatto nel "USER'S MANUAL F8 DESIGN EVALUATIONS KIT NUMBER ONE".

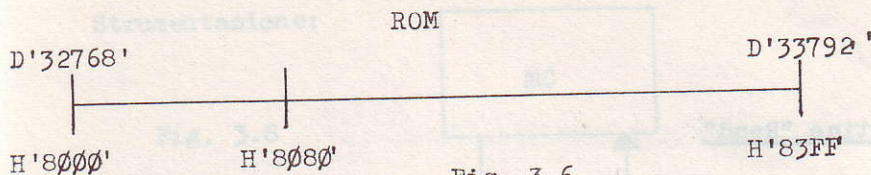



Fig. 3.6

Per eseguire il Debug si può agire in due modi: uno attraverso il programma con l'istruzione PI H'8Ø8Ø, l'altro direttamente attraverso il microcalcolatore mandando lo switch "Debug" a off. Quando si entra in Debug la TTY stampa "?" e da qui segue il comando.

Subroutine "Read"

Inizialmente dobbiamo introdurre il numero massimo dei caratteri della stringa in un registro RX (X per indicare uno dei 64 della CPU).

Inoltre dobbiamo mettere nel Data-Counter l'indirizzo di partenza del Buffer.

Una volta realizzata la Subroutine di "Read" noi avremo tre blocchi a disposizione:

Fig. 3.7

PROG. CHIAMANTE	SUB. PRINT	SUB. READ

Vogliamo strutturare i tre blocchi in modo tale che:

a) Si battono dei caratteri sulla TTY.

b) Di ciascuno di questi caratteri viene stampata subito la "eco".(*)

c) Quando il numero dei caratteri introdotti è uguale al numero massimo, allora si ha la stampa di tutto il buffer (stringa).

(*) Nota

Il programma "eco" è quel programma che ci permette di battere sulla tastiera e leggere immediatamente il carattere voluto. Nella Subroutine "Read" il programma "eco" è costituito dalle tre istruzioni agli indirizzi: 42D, 430, 431.

Strumentazione:

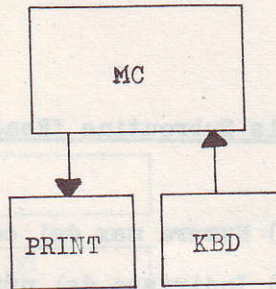
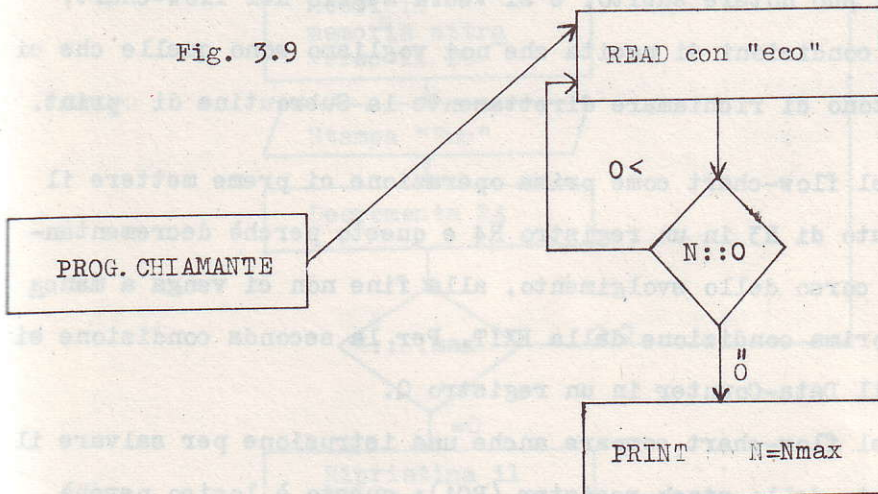
Fig. 3.8

Si può schematizzare quanto detto prima, così:

Fig. 3.9

Realizzazione della Subroutine "Read"

ENTER: 1) Numero max dei caratteri in R3

2) Indirizzo del primo carattere nel DC

3) In RØ il Delay Counter

EXIT: 1) In R3 il numero dei caratteri

2) DC come un ingresso

3) In RØ il Delay Counter

Si può notare subito, e si vedrà meglio nel flow-chart, che le condizioni di uscita che noi vogliamo sono quelle che ci permettono di richiamare direttamente la Subroutine di **print**.

Nel flow-chart come prima operazione ci preme mettere il contenuto di R3 in un registro R4 e questo perchè decrementando nel corso dello svolgimento, alla fine non ci venga a mancare la prima condizione della EXIT. Per la seconda condizione si salva il Data-Counter in un registro Q.

Nel flow-chart compare anche una istruzione per salvare il contenuto dello stack-register (PC1): questo è logico perchè prima di realizzare la Subroutine dobbiamo salvare l'indirizzo dell'istruzione del programma principale, successiva a quella dove abbiamo interrotto.

Alla fine dobbiamo ripristinare i contenuti che abbiamo salvato.

FLOW-CHART

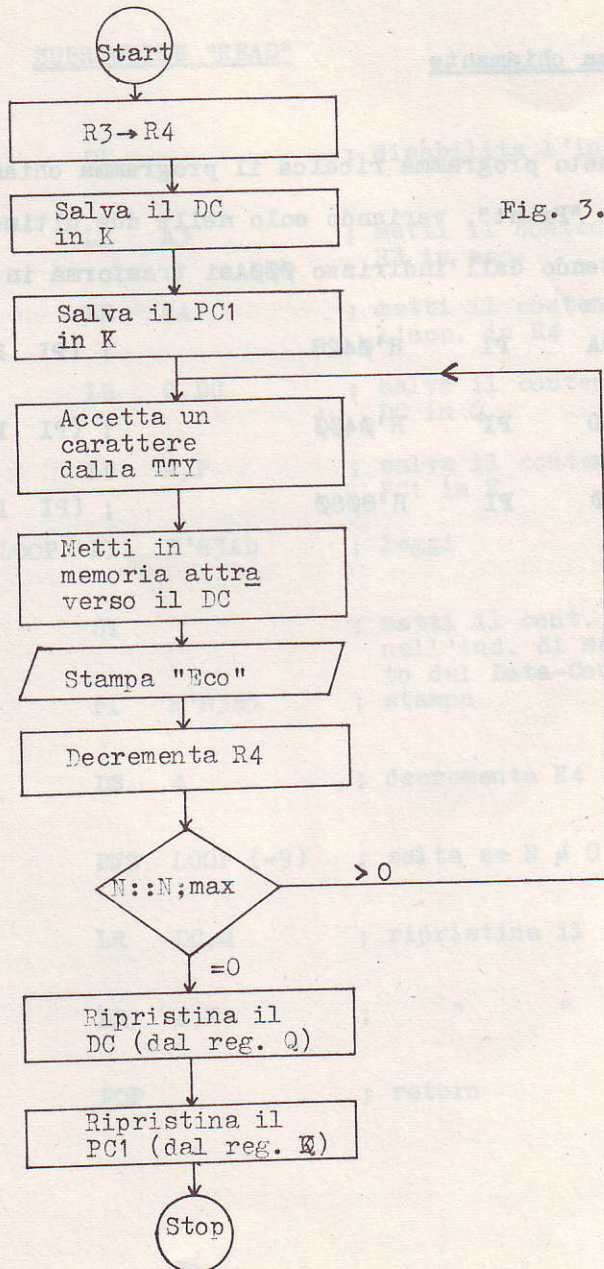



Fig. 3.10

Programma chiamante

Questo programma ricalca il programma chiamante dello Subroutine "Print", variando solo nelle due ultime istruzioni, che partendo dall'indirizzo ØØØAsi trasforma in:

```
ØØØA    PI    H'Ø428         ; (PI READ)
ØØØD    PI    H'Ø4ØØ         ; (PI PRINT)
ØØ1Ø    PI    H'8Ø8Ø         ; (PI DEBUG)
```

SUBROUTINE "READ"

```
428  1A                DI            ; disabilita l'interrupt

429  43                LR   A3       ; metti il contenuto di
                                       R3 in acc.
42A  54                LR   4A       ; metti il contenuto del
                                       l'acc. in R4
42B  ØE                LR   Q,DC     ; salva il contenuto del
                                       DC in Q
42C  Ø8                LR   K,P      ; salva il contenuto del
                                       PC1 in K
42D  28 83 AD   LOOP   PI   H'83AD   ; leggi

43Ø  17                ST            ; metti il cont. dell'acc.
                                       nell'ind. di mem. punta
                                       to dal Data-Counter
431  28 83 E5          PI   H'83E5   ; stampa

434  34                DS   4        ; decrementa R4

        (-9)
435  94 F7             BNZ  LOOP (-9) ; salta se N ≠ 0

437  ØF                LR   DC,Q     ; ripristina il DC

438  Ø9                LR   P,K      ;       "     "  PC1

439  1C                POP           ; return
```

Programma "confronto"

Questo programma è fatto per confrontare l'uguaglianza fra due caratteri e quindi usare l'istruzione di confronto CI.

Particolarmente nel caso del nostro programma vogliamo confrontare il carattere accettato dalla TTY con un carattere particolare CR, cioè ritorno carrello.

Vogliamo dunque che, se battiamo sulla TTY il tasto CR, avvenga qualcosa e precisamente si realizzi la Subroutine TTCR che comporta in uscita dalla TTY la realizzazione dei seguenti passaggi CR/LF/NULL. Con CR si ha il ritorno carrello, con LF (Line Feed) si va a caporigo.

FLOW-CHART

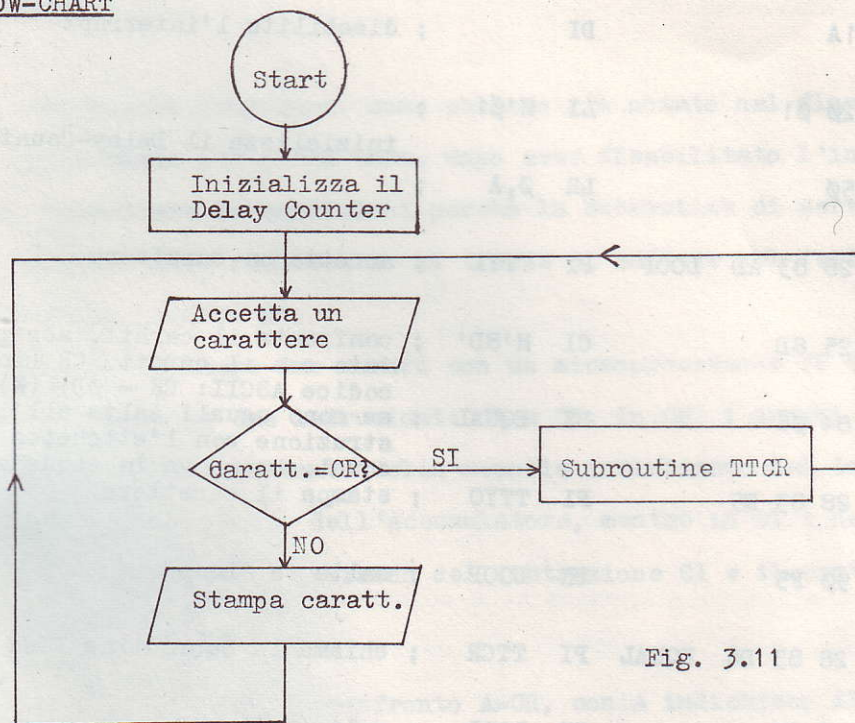



Fig. 3.11

Da notare che il primo blocco significativo si preoccupa di inizializzare il Delay-Counter e questo si fa per soddisfare le condizioni per la realizzazione della TTYI.

PROGRAMMA CONFRONTO

```
ØØØØ  1A               DI          ; disabilita l'interrupt

ØØØ1  2Ø Ø1            LI  H'Ø1    ;
                                     inizializza il Delay-Counter
ØØØ3  5Ø               LR  Ø,A     ;

ØØØ4  28 83 AD  LOOP   PI  TTYI    ; accetta un carattere

ØØØ7  25 8D            CI  H'8D'   ; confronta il caratt. accet
                                     tato con il caratt. CR (in
                                     codice ASCII: CR → ØD) (*)
ØØØ9  84 Ø6            BZ  EQUAL   ; se sono uguali salta all'i
                                     struzione con l'etichetta
                                     equal, se no va in sequenza
ØØØB  28 83 E5         PI  TTYO    ; stampa il carattere

ØØØE  9Ø F5            BR  LOOP    ; salta a "Loop"

ØØ1Ø  28 83 D6  EQUAL  PI  TTCR    ; chiama la Subroutine TTCR

ØØ13  9Ø FØ            BR  LOOP    ; salta a "Loop"
```

(*) Nota

In codice ASCII il carattere CR è ØD mentre nell'operand dell'istruzione CI compare 8D: questo perchè la nostra TTY mette a 1 il primo bit (bit di parità).

Ø	D
ØØØØ	11Ø1
1ØØØ	"
8	D

Commento

Nel nostro programma, come abbiamo già notato nel flow-chart è necessario per prima cosa, dopo aver disabilitato l'interrupt, rispettare le condizioni perchè la Subroutine di servizio TTYI possa essere realizzata: si tratta di mettere nel registro O, il Delay-Counter Ø1.

Per confrontare due numeri con un microprocessore F8 abbiamo a disposizione due istruzioni: CM e CI. In CM i numeri confrontati sono il contenuto della memoria indirizzato dal Data Counter e il contenuto dell'accumulatore, mentre in CI i numeri confrontati sono l'operando dell'istruzione CI e il contenuto dell'accumulatore.

Quando facciamo il confronto A=CR, con A indichiamo il carattere accettato, A si trova in accumulatore.

Come ultima notazione da fare, prendiamo in considerazione le condizioni che la Subroutine di servizio TTCR richiede e lo stato che lascia dopo la sua esecuzione. Come condizioni di ENTER richiede solo il Delay-Counter in RØ e questo lo abbiamo messo all'inizio, inoltre come EXIT presenta il registro RØ invariato e quindi soddisfa l'unica condizione di ENTER della TTYI che può quindi essere richiamata direttamente.

Programma "Terminal"

Il programma "Terminal" serve per creare in memoria (a partire dalla locazione Ø2ØØ) un buffer di lunghezza variabile.

Vengono riconosciute 2 categorie di caratteri:

1) Carattere stampato e messo nel buffer.

2) Caratteri di controllo RUBOUT, "←", CR.

Il RUBOUT serve per cancellare gli ultimi caratteri battuti, mentre il simbolo "←" serve per cancellare tutto il buffer. CR infine serve per far stampare il buffer.

Per fare questo programma si è seguito uno schema che consiste in una prima parte di inizializzazione, una seconda di riconoscimento dei tre caratteri di controllo e infine una terza fase finale che riguarda la stampa del carattere.

Naturalmente se nel corso del programma viene riconosciuto uno dei tre caratteri di controllo, segue una serie di istruzioni appropriate al caso.

Tutto ciò è più evidenziato e specificato nel flow-chart e nel programma.

FLOW-CHART

Start
Condizioni iniziali
Carica DC ind. Buffer
Decrementa DC
Accetta un caratt.
E' RUBOUT?
SI
Stampa "\"
NO
E' "CR"?
SI
Aggiungi CR nel Buffer
Stampa CR, LF, NULL
Rimetti DC=200
NO
Carica da B. car. ind. da DC
E' "←"?
SI
NO
Metti carat nel Buffer
Stampa "←"
E' "CR"?
SI
NO
Stampa il carattere
Esegui "TTCR"
Incrementa il DC (aut.)
Incrementa il DC (aut.)
Stampa il carattere
Stop

PROGRAMMA TERMINAL

```
ØØØØ  1A                 DI                ; disabilita l'inter
                                             rupt
ØØØ1  2Ø Ø1              LI    H'Ø1        ;
                                             inizializza il De
                                             lay-Counter
ØØØ3  5Ø                 LR    Ø,A         ;
ØØØ4  2A Ø2 ØØ  START    DCI   H'Ø2ØØ      ; inizio del buffer
ØØØ7  28 83 AD  LOOP     PI    TTYI        ; accetta un carat-
                                             tere
ØØØA  25 FF              CI    H'FF'       ; è un "rubout" se
                                             sì vai all'etichet
                                             ta " Not Equal"
ØØØC  94 ØC              BNZ   NOT EQUAL   ;
ØØØE  2Ø DC              LI    H'DC'       ;
ØØ1Ø  51                 LR    1,A         ; stampa " \ "
                                             (= H'DC')
ØØ11  28 83 E5           PI    TTYO        ;
ØØ14  2Ø FF              LI    H'FF4       ; -1 sull'accumulato
                                             re
ØØ16  8E                 ADC               ; somma -1 al DC
ØØ17  9Ø EF              BR    LOOP        ; torna ad accettare
                                             un altro carattere
```

(continua)

Handwritten notes: "Δ4 x300 baud", "Ø6 x110", "7F" (next to 25 FF and H'FF').

```
ØØ19  25 8D    NOT EQUAL  CI   H'8D'     ;
                                            è un "CR" se si vai
                                            all'etichetta "CR"
ØØ1B  84 11               BZ   CR        ;
ØØ1D  25 DF               CI   H'DF'     ; è un "←" ?
ØØ1F  94 Ø7               BNZ  CONTINUE  ; se no vai all'eti
                                            chetta "continue"
ØØ21  51                  LR   1,A       ; se si stampa "←"
ØØ22  28 83 E5            PI   TTYO      ;
                                            ricomincia da capo
ØØ25  9Ø DE               BR   START     ;
ØØ27  17       CONTINUE   ST             ; se non era "←",
                                            "RUBOUT", "CR"
ØØ28  28 83 E5            PI   TTYO      ; stampa il caratte
                                            re
ØØ2B  9Ø DB               BR   LOOP      ; vai ad accettarne
                                            un altro
ØØ2D  17       CR         ST             ; se era CR, memoriz
                                            za CR nel buffer
ØØ2E  28 83 D6            PI   TTCR      ; stampa CR, LF, NULL
ØØ31  2A Ø2 ØØ            DCI  H'Ø2ØØ'   ; resetta il Pointer
                                            all'inizio del buf
                                            fer
ØØ34  16       LOOP1      LM             ; carica la locazio
                                            ne puntata dal DC
ØØ35  25 8D               CI   H'8D'     ; è un CR ?
```

(continua)

```
ØØ37  84 Ø7                BZ  STOP    ; se si vai all'istru
                                          zione di stop
ØØ39  51                   LR  1,A     ;
                                          se no stampa il ca
                                          rattere
ØØ3A  28 83 E5             PI  TTYO    ;

ØØ3D  9Ø F6                BR  LOOP1   ; torna all'etichetta
                                          Loop1

ØØ3F  28 83 D6  STOP       PI  TTCR    ; se è CR stampa CR,
                                          LF, NULL

ØØ42  9Ø FF     SELF       BR  SELF    ; fermati
```

29 00 00

Da notare che i caratteri in esadecimale DC', 8D', DF' stanno a rappresentare rispettivamente i simboli: "\", "CR", "←".

La sbarretta, che sappiamo cancella l'ultimo carattere battuto, si usa così, per esempio:

```
    Si batte    FRASE       QUALUNC\QUE      (Si batte CR)
                FRASE       QUALUNQUE
```

CAPITOLO 5

Tecniche Input/Output

Il trasferimento dei dati fra un sistema microcomputer F8 e il mondo esterno può avvenire in tre modi:

a) I/O Programmato

b) Interruzione I/O

c) DMA (Accesso diretto alla Memoria)

I/O Programmato

Questo sistema di trasferimento dati è caratterizzato dall'esecuzione da parte della CPU di un'istruzione che inizializza e controlla il trasferimento di dati, attraverso una porta di ingresso/uscita.

Si hanno 4 istruzioni che abilitano l'I/O programmato: INS, IN, OUTS e OUT.

Inoltre si può dire che non tutte le porte I/O che abbiamo a disposizione possono essere usate per il trasferimento dei dati, infatti alcune di queste servono per il controllo del. sistema d'interruzione del programma e dei clock programmabili.

Il modo più semplice di usare l'I/O programmato è di far sì che l'unità esterna mandi in ingresso un segnale quando è

pronta a trasmettere o a ricevere dati: questo sistema è chiamato "Polling on status", ossia "interrogazione sullo stato".

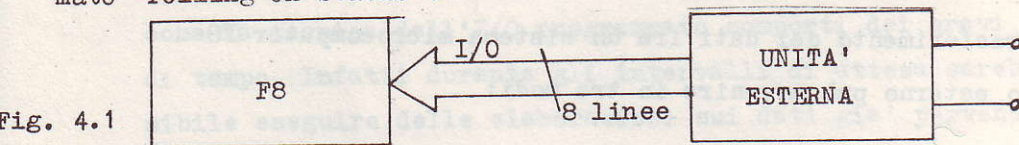
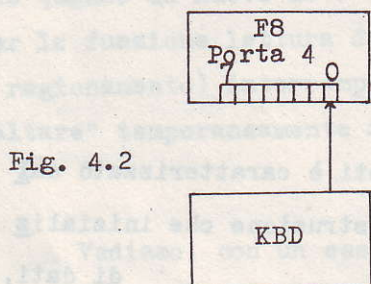
Fig. 4.1

Per fare un esempio pratico vediamo cosa succede quando sulla tastiera di una TTY battiamo un tasto: il carattere da noi battuto è tradotto in 8 bit che verranno inviati all'F8in serie.

Fig. 4.2

Oltre gli 8 bit che compongono il caratterevengono inviati un bit di START e due bit di STOP.

Il bit di START rappresenta, nel nostro caso, il segnale di inizio trasmissione ed è possibile verificare lo stato della periferica con la stessa linea con la quale si ricevono i dati.

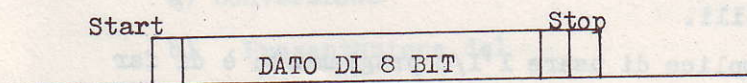



Fig. 4.3

Interruzione I/O

Questo metodo di trasferimento di dati sarà analizzato più a fondo perchè verrà poi usato nelle applicazioni.

Le interruzioni sono in genere usate nel controllo delle operazioni di I/O per due motivi particolari: per eliminare il tempo perso nell'esaminare lo stato della macchina (Polling on status) e per trattare gli eventi I/O imprevedibili.

Lo svantaggio considerevole, che L'I/O programmato comporta, è dovuto al fatto che il microprocessore impiega la maggior parte del suo tempo nell'attesa del byte di controllo. Riprendendo l'esempio fatto prima si può vedere cosa fa il microcomputer prima di eseguire il programma per accettare un dato.

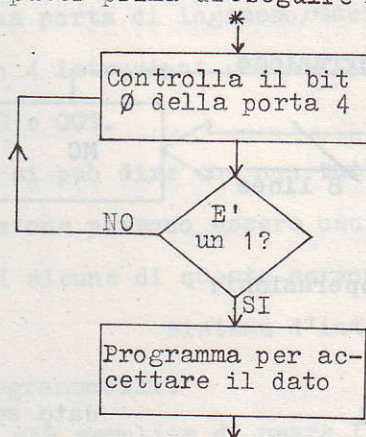



Fig. 4.4

Come si vede il microcomputer prima che gli arrivi il byte di controllo continua a fare dei cicli, senza eseguire alcuna operazione utile.

Se si considera che la CPU in un secondo e' in grado di eseguire svariate migliaia di istruzioni, si comprende bene come la tecnica dell'I/O programmato comporti dei gravi sprechi di tempo. Infatti durante gli intervalli di attesa sarebbe possibile eseguire delle elaborazioni sui dati gia' pervenuti fino a quel momento.

La tecnica dell'interruzione invece permette di far funzionare normalmente il microcomputer per le elaborazioni, e solo quando un nuovo dato e' pronto sulla porta di ingresso (per la funzione lettura dati, per la scrittura basta invertire il ragionamento) interrompere il programma di elaborazione e "saltare" temporaneamente al programma che regola l'ingresso dati.

Vediamo, con un esempio, come ciò si realizza: supponiamo di avere un convertitore analogico/digitale (A/D) e un microcomputer.

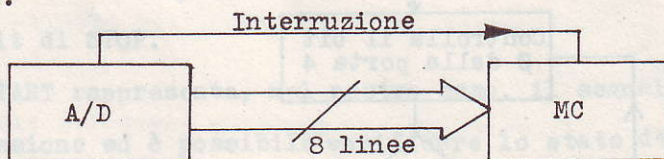



Fig. 4.5

L'A/D fa le seguenti operazioni:

a) Conversione

b) Presentazione del dato sulla porta di ingresso/uscita

c) Manda un segnale di interruzione

Il microcomputer esegue l'elaborazione dei dati mentre l'A/D effettua la conversione e la presentazione.

Quando arriva il segnale di interruzione, passa in esecuzione un nuovo programma, e viene sospesa momentaneamente l'elaborazione principale. Questo programma può essere composto dalle seguenti istruzioni:

1) Salva lo stato della macchina (Accumulatore, registri, ecc.).
2) Accetta un dato dalla porta collegata all'A/D.
3) Memorizza.
4) Ripristina lo stato.
5) Ritorna al punto dove il programma principale è stato interrotto.

Se avessimo usato il "Polling on status" tutto il tempo sarebbe andato perso nell'attesa di dati dalla unità A/D.

Il trasferimento di una serie di bytes di dati a una frequenza conosciuta costituisce una serie di eventi prevedibili.

In molte applicazioni la necessità che una unità esterna debba accedere al sistema microprocessore non può essere prevista. Per esempio un'unità esterna può comunicare col microcomputer solamente in circostanze stringenti; in questi tempi il microcomputer deve eseguire un programma per calcolare e far uscire le necessarie correzioni dei dati.

Quando la necessità che un'unità esterna debba accedere al microcomputer non può essere prevista, un'interruzione è il solo modo ragionevole con il quale l'unità esterna può ottenere il controllo del sistema microprocessore.

Facciamo un esempio anche di questo caso.

Supponiamo di avere un microcomputer che accetta la temperatura di un forno quando si preme un certo carattere su una telescrivente per esempio "↑". Il processo di cambiamento della temperatura inizia quando si batte, per esempio, "CR".

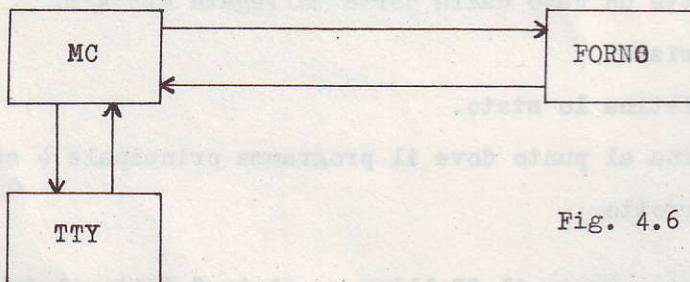



Fig. 4.6

Con il Polling dopo aver battuto "↑", se per una qualsiasi ragione non viene battuto "CR", il programma rimane bloccato in un LOOP e la temperatura del forno sale, rischiando, per esempio, gravi danni.

Invece con il sistema delle interruzioni, una volta battuto "↑", anche se non si batte il "CR", se un apposito sensore segna la temperatura pericolosa e genera un segnale di interruzione puo' essere eseguito un programma "emergenza" limita la temperatura.

Riassumendo si può dire che mentre nel'I/O programmato il trasferimento dati è comandato dal programma, attraverso l'interruzione I/O sono le unità esterne che comandano il trasferimento.

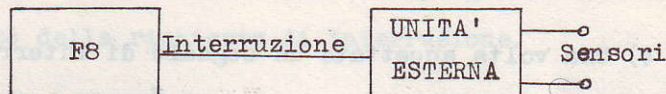



Fig. 4.7

<u>Sequenza di eventi alla richiesta di una interruzione</u>

La sequenza di eventi, che si succedono nel microcomputer F8, quando giunge una richiesta di interruzione, è la seguente:

1) Alla ricezione del segnale di interruzione, il sistema di interruzione deve essere abilitato dalla CPU: tutte le interruzioni sono abilitate o disabilitate attraverso la CPU. Una volta che la CPU ha abilitato un'interruzione, ciascun chip 3851 PSU e 3853 SMI, tramite la propria linea individuale di interruzione, potrà o meno abilitare o disabilitare l'interruzione.

2) Può accadere che più richieste di interruzione arrivino simultaneamente alla CPU, (richieste da più unità esterne): in questo caso è stabilita una priorità di cui parleremo più avanti.

3) Quando la 3850 CPU riceve una richiesta di interruzione, viene sospesa l'esecuzione del programma corrente alla fine dell'istruzione che viene eseguita in quel momento. Ci sono però 8 istruzioni alla fine delle quali non viene riconosciuta

l'interruzione: queste sono PK; PI; POP; JMP; OUTS; OUT; EI; LRW,J. Quindi un'altra istruzione, escluse quelle sopra, deve essere eseguita prima che un'interruzione sia riconosciuta.

4) Una volta accettato un segnale di interruzione, la CPU manda fuori un segnale di riconoscimento ed è proprio il modo con cui questo segnale è inviato che determina la priorità delle interruzioni in presenza di più richieste simultanee.

5) Quando la CPU manda fuori il segnale di riconoscimento di un'interruzione, lo stato che abilità l'interruzione all'interno della CPU è azzerato, disabilitando così ogni interruzione che segue. Le interruzioni devono essere riabilitate sotto controllo del programma.

6) Ciascun chip oltre ad una linea di richiesta di interruzione ha anche un registro indirizzo a 16 bit che contiene l'indirizzo della prima istruzione da eseguire dopo l'interruzione. Il registro indirizzo 3851 è realizzato direttamente in fabbrica, programmabile a richiesta, mentre il registro indirizzo della 3853 è costituito da due porte di I/O che sono caricate da una appropriata istruzione. Il chip che riceve il segnale di riconoscimento dell'interruzione risponde trasmettendo il contenuto del suo registro indirizzo dell'interruzioni.

7) La logica della PSU manda il contenuto del PCO nel PC1 e carica l'indirizzo della prima istruzione da eseguire dopo l'interruzione nel PCØ così da eseguire il programma dedicato al riconoscimento della richiesta di interruzione.

Ricordiamo infine che nell'accensione del sistema tutte le interruzioni sono disabilitate.

Abilitazione e disabilitazione delle interruzioni

Per abilitare la richiesta di interruzione, la CPU si serve del bit 4 (ICB) del registro W, quando ICB = 1, la richiesta di interruzione alla CPU è abilitata, quando ICB = Ø, le richieste di interruzione non sono riconosciute: ICB è posto uguale a 1 dall'istruzione EI, è azzerato dall'istruzione DI.

Per quanto riguarda gli altri chip le richieste individuali di interruzione sono controllate attraverso una porta I/O: le seguenti due istruzioni caricano una porta di controllo dell' interruzione.

```
LI   VAL
OUT  IPRT
```

IPRT deve essere uguagliato all'indirizzo della porta I/O di controllo dell'interruzione nel chip in esame e VAL deve essere uguale a quanto segue:

VAL

H'00' Interruzioni disabilitate nel chip che interessa

H'01' " esterne abilitate

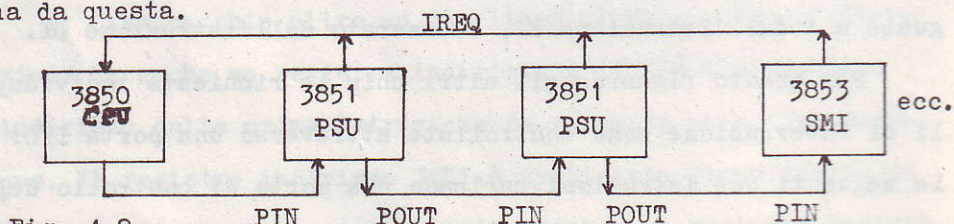
H'02' " disabilitate nel chip che interessa

H'03' " esterne disabilitate

Priorità delle istruzioni

Quando un microcomputer F8 ha più di una linea di interruzioni le priorità sono determinate con la tecnica "daisy chaining", come riportato in figura 4.8.

Come si vede dalla figura la priorità maggiore spetta alla CPU e si va via via decrescendo man mano che ci si allontana da questa.

Fig. 4.8

IREQ è la linea comune di richiesta di interruzione

PIN la priorità IN (Interruzione riconosciuta) e POUT la priorità OUT.

Ciascuna unità, quando gli arriva PIN, passa alle altre unità il segnale POUT, a meno che non vi sia una richiesta di

interruzione, nel qual caso viene eliminato il PIN.

Il chip 3853 SMI non dispone del segnale di uscita POUT perciò esso sarà l'ultima della catena e avrà la più bassa priorità di interruzioni.

Una volta che abbiamo stabilito fisicamente le priorità, la sola cosa che un programmatore può fare per modificare questa scelta è disabilitare le interruzioni esterne su chip selezionati, caricando appropriatamente la porta di controllo.

Quando la CPU ha riconosciuto l'interruzione e il bit ICB del registro W è stato posto a 1, entra in funzione la routine di servizio dell'interruzione, che può essere anch'essa interrotta.

Per evitare quest'ultima possibilità, il bit ICB deve essere azzerato fino a che la routine di servizio dell'interruzione ha completato l'esecuzione.

Vediamo tramite una figura cosa succede nel caso di arrivo di due interruzioni consecutive.

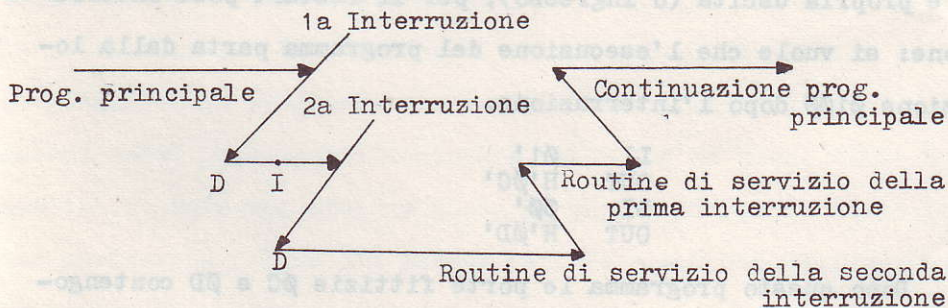



Fig. 4.9

Con D si indica che le interruzioni sono disabilitate, quindi ICB=O, mentre I vuol dire che le interruzioni sono abilitate, quindi ICB=1.

L'abilitazione dei singoli chips e la logica del programma (non la priorità delle interruzioni) determinano qual'è la prima interruzione e quale la seconda.

All'interno della CPU dopo il riconoscimento di una interruzione si procede esattamente come se si fosse chiamata una subroutine: il contenuto di PCO passa in PC1 e il contenuto del registro indirizzo post-interruzione del chip selezionato passa in PCO. Le interruzioni sono perciò trattate come se fosse richiesta una subroutine. Ritornare da un'interruzione al programma interrotto è lo stesso che ritornare da una subroutine al programma chiamante.

Per concludere si può vedere un uso delle porte I/O, ØC e ØD, che in effetti sono porte fittizie perchè non sono una vera e propria uscita (o ingresso), per il restart post-interruzione: si vuole che l'esecuzione del programma parta dalla locazione Ø1ØØ dopo l'interruzione.

```
LI    Ø1'
OUT   H'ØC'
LI    ØØ'
OUT   H'ØD'
```

Dopo questo programma le porte fittizie ØC e ØD contengono l'indirizzo Ø1ØØ che serve per l'interruzione. Quando arriva un'interruzione l'esecuzione procede da Ø1ØØ.

DMA (Direct Memory Access)

L'Accesso Diretto alla Memoria permette che i dati siano trasferiti fra la memoria di un sistema microcomputer F8 e una unità esterna, senza passare per la CPU: i dati sono trasferiti in parallelo alle operazioni della CPU.

Un chip 3852 DMI deve essere presente in un sistema microcomputer per fare da supporto al DMA. Nel sistema possono essere presenti non più di 4 chips 3854 DMA, ciascuno dei quali fornisce un canale DMA.

Maggiori particolari sul DMA, sono dati nel "A guide to programming the Fairchild F-8 microcomputer".

CAPITOLO 6

Il Child 8/BS

L'impostazione del CHILD/BS ricalca essenzialmente quella suggerita dalla Fairchild per il Kit N°1,ma vi apporta delle migliorie che ritengo veramente determinanti ai fini della flessibilità di impiego e della possibilità di espanzione.La CPU costituisce come ovvio il "cervello" di tutto il sistema mentre i clocks ed il control bus regolano il funzionamento di tutto il circuito.La PSU mette a disposizione 1 K bytes di memoria ROM (Fair-Bug) con indirizzo H'8000'-H'83FF' (D'32768'-D'33792'). Inoltre la PSU fornisce due port di I/O,numerati 4e 5 che si aggiungono allo Ø e all'1 presenti sulla CPU,un livello di interrupt (quello a priorità maggiore) ed un timer programmabile. Sul port numero 4 è collegata l'interfaccia per la telescrivente ASCII 7+1+3 bit per velocità di 110-300 baud.La tabella 1 riporta le principali caratteristiche di alcune delle subroutines (sottoprogrammi,ossia programmi di servizio richiamabili da altri programmi detti principali) contenute sulla PSU ed assai utili in pratica.Il Fair-Bug dispone anche di programmi per il controllo di una periferica veloce con uscita parallela a 8 bit come un lettore ottico di nastro perforato.

Tabella N°1

Nome Simbolico	Indirizzo esadec.	Funzione
TTYI	83AD	Accetta un carattere ASCII dalla tastiera e pone il suo codice nello accumulatore
TTYO	83E5	Provoca la stampa sulla telescrivente del carattere ASCII il cui codice esadecimale si trova nel registro numero 1
TTCR	83D6	Fa tornare a capo e a nuova riga il carrello della telescrivente.

La SMI infine consente di collegare memoria esterna di tipo statico oltre a disporre di un livello di interrupt (il più basso) e di un timer programmabile che si aggiunge a quello presente sulla PSU. Sulla scheda CPU del Child si trova inoltre 1 K bytes di memoria RAM (otto integrati 2102-2).

Si ha poi il DEBUG il cui scopo è quello di regolare il funzionamento perchè alla pressione del tasto RESET sul pannello di comando l'esecuzione possa procedere o dalla locazione H'0000' oppure dalla H'8080' (inizio Fair-Bug).

Infine ci sono i dispositivi per il pilotaggio delle linee esterne che escono dal connettore per fornire loro la potenza necessaria a pilotare altre schede. Il data bus in particolare, essendo bidirezionale, richiede anche dei circuiti che stabiliscano, in funzione di certi segnali, la direzione del flusso di informazioni.

CAPITOLO 7

ESEMPIO DI APPLICAZIONE PRATICA

Programma di acquisizione A/D su interruzione - Port. N°1

Questo programma, che in seguito chiameremo AAD (Acquisizione Analogica Digitale), è stato realizzato per permettere l'acquisizione da parte del microcalcolatore CHILD 8/BS di un insieme di dati, e precisamente tre blocchi di 256 campioni, provenienti dal convertitore A/D , attraverso una serie di interruzioni successive.

Con il programma AAD è stata dunque completata l'acquisizione di un segnale analogico da parte del microcalcolatore CHILD 8/BS.

Le parti che compongono il programma AAD contengono una inizializzazione delle interruzioni e una inizializzazione di un contatore, segue poi la memorizzazione dei dati una volta arrivato il segnale di interruzione e infine la stampa di un simbolo, nel nostro caso @ (H'40' in codice ASCII), per attestare il riconoscimento del segnale di interruzione.

Vediamo tutto ciò più specificatamente prima con un Flow-chart, poi con la stesura del programma.

<u>FLOW-CHART</u> (Prog. AAD)

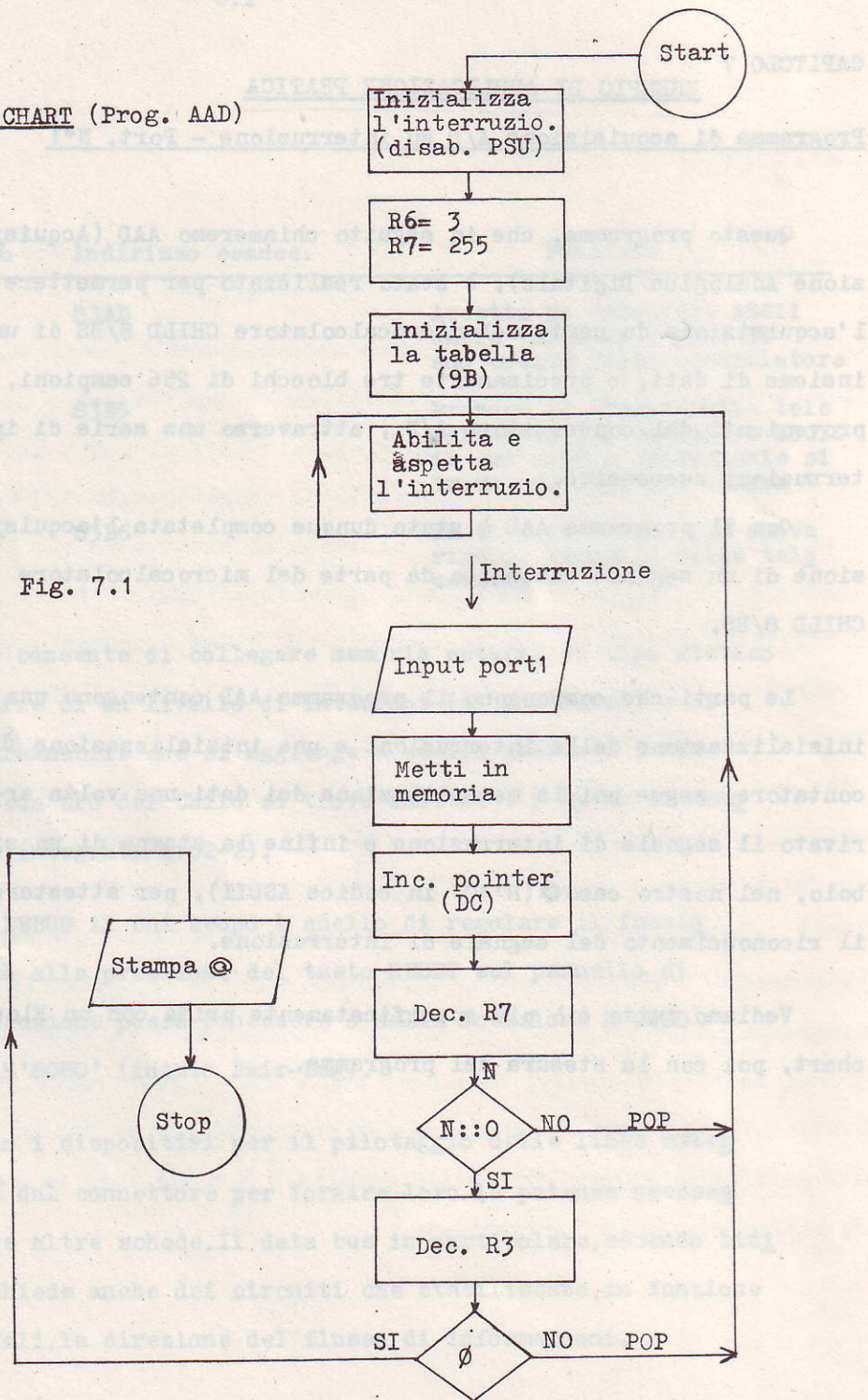



Fig. 7.1

PROGRAMMA "AAD"

```
ØØØØ  1A         INIZ  DI          ; disabilita le interru-
                                     zioni (nella CPU)
                                     ICB=0
   1  7Ø               LIS Ø       ;
                                     disabilita l'inter. nel
                                     la PSU
   2  B6               OUTS Ø6     ;

   3  71               LIS 1       ;
                                     abilita l'inter. nella
                                     SMI
   4  BE               OUTS E      ;

   5  7Ø               LIS Ø       ;
                                     metti la parte alta del
                                     vettore di inter. =0
   6  BC               OUTS C      ;

   7  2Ø 96            LI 96       ;
                                     carica la parte bassa
   9  BD               OUTS D      ;


   A  2Ø FF      CONT  LI FF       ;
                                     carica R7 con FF
   C  57               LR 7,A      ;

   D  2Ø Ø3            LI Ø3       ;
                                     carica R6 con 3
   F  56               LR 6,A      ;

  1Ø  2A ØØ 93         DCI TABLE   ; metti nel DC l'ind.
                                     della Tabella
```

(continua)

```
ØØ13  1B        SELF   EI          ; abilita le interruzioni
                                     (nella CPU) ICB=1

  14  9Ø FE            BR SELF     ; attesa del segnale di
                                     interruzione

  16  A1        INTER  INS 1       ; metti nell'Acc. il con
                                     tenuto del Port1

  17  17               ST          ; metti in memoria

  18  37               DS 7        ;

  19  84 Ø2            BZ DEC      ; decrementa R7 e torna
                                     se non contiene 0

  1B  1C               POP         ;

  1C  36        DEC    DS 6        ;

  1D  84 Ø2            BZ END      ; decrementa e torna se
                                     R6 non contiene 0

  1F  1C               POP         ;


  2Ø  2Ø 4Ø     END    LI H'40'    ;

  22  51               LR 1,A      ;
                                     stampa@ e Stop
  23  28 83 E5         PI TTYO     ;

  26  9Ø FF     SELF1  BR SELF1    ;

  96  29 ØØ 16         JMP 16      ; salta all'ind. 16

  9B            TABLE              ; tabella dei dati
```

Note

Il vettore di interruzione ha 16 bit per poter ripartire da una qualunque delle locazioni di memoria ($2^{16} = 65536$)

Il bit N° 7 è fisso a:

1 - interruzione esterna (*)

Ø - timer

Nel caso (*) quindi bisogna tener conto che in ottava posizione ci deve essere un 1. Indirizzi, come per esempio H'16' non sono possibili.

Si usa pertanto, se è necessario ricorrere a locazioni di memoria non altrimenti indirizzabili, un salto indiretto (JMP o BR).

E' molto importante notare che dopo ogni interruzione ci deve essere l'istruzione EI, cioè si devono riabilitare le interruzioni.

Per concludere si può dire che si è abilitato la SMI e non la PSU, perchè quest'ultima richiede un vettore fisso di interruzione che cade in mezzo alla memoria, mentre la SMI ha un vettore di interruzione programmabile.

APPENDICE A

Sistema di numerazione binario

In genere, nei nostri calcoli, si considera un sistema di numerazione che utilizza 10 simboli, i numeri da 0 a 9, mentre gli elaboratori elettronici possono conprendere solo il sistema binario, che comprende solo numeri costituiti da due sole cifre 0 e 1: queste due cifre possono rappresentare due livelli di tensione 0=0 volt; 1=+5 volt. Le cifre 0 e 1 sono dette bit.

La base del sistema di numerazione binario, come dice il nome stesso, è 2. Consideriamo per esempio un numero binario:

0 1 1 0 1 0	Numero binario
5 4 3 2 1 0	Indice di posizione
$2^5 2^4 2^3 2^2 2^1 2^0$	Potenze di base 2
32,16,8,4,2,1	Valore in base decimale

L'equivalente decimale di 011010 è:

$0x2^5+1x2^4+1x2^3+0x2^2+1x2^1+0x2^0 = 16+8+2+0 = 26$

Si possono vedere gli algoritmi per il passaggio da un sistema di rappresentazione all'altro, analizzando due esempi:

1)	1	0	1	0	1	In decimale equivale a 21
	1x2=2	0+ 2= 2	2x2=4 1+ 4= 5	5x2=10 0+ 10= 10	10x2=20 1+ 20= 21	

Cioè moltiplico per 2 e sommo 0 o 1 a seconda di quello che trovo.

2) Numero in decimale Resto

		Resto
26 !	2	0
13 !	2	1
6 !	2	0
3 !	2	1
1 !		1

Il numero in binario è 11010

Per scrivere in binario un numero non intero utilizzo le potenze del 2 a esponente negativo.

Per esempio il numero 25,25 è equivalente, in binario a:

011001.1 $0x2^5+1x2^4+1x2^3+0x2^2+0x2^1+1x2^0+1x2^{-1}$ = 16+8+1+0,25= = 25,25

Sistemi di numerazione ottale e esadecimale

Nei calcolatori sono molto in uso anche numerazioni diverse da quella binaria o decimale: si hanno sistemi di numerazione ottale e esadecimale:

Il primo è un sistema a base 8 e può essere ricavato direttamente dal sistema binario raggruppando i bit a gruppi di 3, per esempio: Base 2 010 111 011 110 = 0'82736'

" 8 2 7 3 6

Il secondo è un sistema a base 16 analogamente a prima, si considerano i numeri a gruppi di 4, per esempio:

Base 2 1101 1111 0001 1010 = H'DF1A'

" 16 D F 1 A

E' pratica comune far precedere il numero scritto in ottale dal simbolo @ (oppure O')e scritto in esadecimale dal simbolo H'.

La scelta tra l'utilizzazione del sistema ottale o esadecimale dipende solamente dal numero di bit con cui si ha a che fare: se si ha 12 bit è comodo dividerli a gruppi di tre e quindi di usare il sistema ottale, se abbiamo 16 bit è comodo il sistema esadecimale.

Numeri binari, decimali, esadecimali e ottali

BINARI	DECIMALI	ESADECIMALI	OTTALI
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	10
1001	9	9	11
1010	10	A	12
1011	11	B	13
1100	12	C	14
1101	13	D	15
1110	14	E	16
1111	15	F	17

Nel corso delle nostre applicazioni sarà molto utile ave-re presente i numeri negativi in esadecimale.

Tabella numeri negativi in esadecimale

D'	H'	D'	H'	D'	H'
-1	FF	-16	F0	-31	E1
-2	FE	-17	EF	-32	E0
-3	FD	-18	EE	-33	DF
-4	FC	-19	ED	-34	DE
-5	FB	-20	EC	-35	DD
-6	FA	-21	EB	-36	DC
-7	F9	-22	EA	-37	DB
-8	F8	-23	E9	-38	DA
-9	F7	-24	E8	-39	D9
-10	F6	-25	E7	-40	D8
-11	F5	-26	E6	-41	D7
-12	F4	-27	E5	-42	D6
-13	F3	-28	E4	-43	D5
-14	F2	-29	E3	-44	D4
-15	F1	-30	E2	-45	D3

-46	D2	56	C8	67	BD	65	AF
-47	D1	57	C7	68	BC	64	E
-48	D0	58	C6	69	BB	63	D
49	CF	59	C5	70	BA	62	C
50	CE	60	C4	71	B9	61	B
51	CD	61	C3	72	B8	60	AA
52	CC	62	C2	73	B7		9
53	CB	63	C1	2	B6		8
54	CA	64	C0	1	B5		7
55	C9	65	BF	70	B4		6
		66	BE	69	3		5
				8	2		
				7	1		
				6	B0		

Prima di passare alla descrizione delle operazioni algebriche e logiche con i bit, diamo la definizione di byte: un byte è un insieme di 8 bit che può rappresentare 256 (2^8) possibili combinazioni di 8 cifre.

Somma algebrica fra numeri binari

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 0 con il resto di 1

Differenza algebrica fra i numeri binari

Qui si richiede il concetto di complemento a due di un numero: il complemento a due di un numero si ottiene complementando il numero (si rimpiazza 0 con 1 e 1 con 0) e aggiungendo 1 al complemento.

Per esempio: il complemento a 2 di 0101 (=5) è

```
 1010+
    1
 ----
 1011
```

Se dobbiamo fare 7-5, in binario è:

```
            0111+
            1011
            ----
riporto 1   0010
```

Per la moltiplicazione e la divisione dei numeri binari si usano rispettivamente le operazioni di differenza e di somma.

Somma logica OR:

$0+0 = 0$

$0+1 = 1$

$1+0 = 1$

$1+1 = 1$

Prodotto logico AND:

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Somma logica OR exclusivo:

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

CODICE _ ASCII

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
NULL	00
SOM	01
EOA	02
EOM	03
EOT	04
WRU	05
RU	06
BELL	07
FE	08
H. Tab	09
Line Feed	0A
V. Tab	0B
Form	0C
Return	0D
SO	0E
SI	0F
DCO	10
X-On	11
Tape Aux. On	12
X-Off	13
Tape Aux. Off	14
Error	15
Sync	16
LEM	17
S0	18
S1	19
S2	1A
S3	1B
S4	1C
S5	1D
S6	1E
S7	1F

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
ACK	7C
Alt. Mode	7D
Rubout (Delete)	7F
!	21
"	22
#	23
$	24
%	25
&	26
'	27
(	28
)	29
*	2A
+	2B
,	2C
-	2D
.	2E
/	2F
:	3A
;	3B
<	3C
=	3D
>	3F
?	3F
[	5B
\	5C
]	5D
↑	5E
← (clear)	5F
@	40
blank	20
0	30

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
A	41
B	42
C	43
D	44
E	45
F	46
G	47
H	48
I	49
J	4A
K	4B
L	4C
M	4D
N	4E
O	4F
P	50
Q	51
R	52
S	53
T	54
U	55
V	56
W	57
X	58
Y	59
Z	5A

INDICE